Contextualisation d'un détecteur de piétons : application à la surveillance d'espaces publics

Thierry Chesnais

N° d'ordre : D.U. 2362
EDSPIC : 614

Université Blaise Pascal - Clermont II

École Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse

Présentée par :

Thierry Chesnais

pour obtenir le grade de

Docteur d'Université

Spécialité : Vision pour la robotique

Contextualisation d'un détecteur de piétons : Application à la surveillance d'espaces publics

---

Soutenue le 24 juin 2013

Composition du jury

M. Roland Chapuis	Président
Mme. Saïda Bouakaz	Rapporteur
M. Arnaud De la Fortelle	Rapporteur
M. Xavier Clady	Examinateur
M. Nicolas Allezard	Encadrant
M. Yoann Dhome	Encadrant, Invité
M. Thierry Chateau	Directeur de thèse

Thèse préparée au sein du Laboratoire Vision et Ingénierie des Contenus du CEA, LIST (Saclay) en collaboration avec l'Institut Pascal (Clermont-Ferrand)

Remerciements

Une thèse est un effort collectif et il est utopique de croire qu'un étudiant solitaire puisse s'épanouir et faire un travail original. Je tiens donc ici à adresser mes remerciements à toutes les personnes qui ont participé de près ou de loin à cet ouvrage.

Ce travail est tout d'abord le fruit d'une collaboration entre le Laboratoire Vision et Ingénierie des Contenus (LVIC) du CEA, LIST (Saclay) et l'Institut Pascal (Clermont-Ferrand). Je tiens donc à remercier ici Messieurs François Gaspard et Michel Dhome pour m'avoir accueilli dans leurs unités.

Je voudrais ensuite remercier Roland Chapuis d'avoir accepté de présider mon jury de thèse, ainsi que Saïda Bouakaz, Arnaud de La Fortelle et Xavier Clady pour m'avoir fait l'honneur d'évaluer mes travaux de recherche.

Je souhaite aussi exprimer ma reconnaissance à mon directeur de thèse Thierry Chateau et à mes encadrants Nicolas Allezard et Yoann Dhome pour tous les conseils prodigués au cours de ces années.

J'aimerais également remercier tous mes collègues et particulièrement les doctorants avec qui j'ai pu échanger et discuter. Sans votre bonne humeur et vos remarques cette thèse n'aurait certainement pas été la même.

Enfin je n'oublie pas les personnes de l'ombre, famille et amis pour leurs encouragements et toute l'aide qu'ils m'ont apportée.

Résumé

La démocratisation de la « vidéosurveillance intelligente » nécessite le développement d'outils automatiques et temps réel d'analyse vidéo. Parmi ceux-ci, la détection de piétons joue un rôle majeur car de nombreux systèmes reposent sur cette technologie. Les approches classiques de détection de piétons utilisent la reconnaissance de formes et l'apprentissage statistique. Elles souffrent donc d'une dégradation des performances quand l'apparence des piétons ou des éléments de la scène est trop différente de celle étudiée lors de l'apprentissage.

Pour y remédier, une solution appelée « contextualisation du détecteur » est étudiée lorsque la caméra est fixe. L'idée est d'enrichir le système à l'aide d'informations provenant de la scène afin de l'adapter aux situations qu'il risque de fréquemment rencontrer.

Ce travail a été réalisé en deux temps. Tout d'abord, l'architecture d'un détecteur et les différents outils utiles à sa construction sont présentés dans un état de l'art. Puis la problématique de la contextualisation est abordée au travers de diverses expériences validant ou non les pistes d'amélioration envisagées. L'objectif est d'identifier toutes les briques du système pouvant bénéficier de cet apport afin de contextualiser complètement le détecteur.

Pour faciliter l'exploitation d'un tel système, la contextualisation a été entièrement automatisée et s'appuie sur des algorithmes d'apprentissage semi-supervisé. Une première phase consiste à collecter le maximum d'informations sur la scène. Différents oracles sont proposés afin d'extraire l'apparence des piétons et des éléments du fond pour former une base d'apprentissage dite contextualisée. La géométrie de la scène, influant sur la taille et l'orientation des piétons, peut ensuite être analysée pour définir des régions, dans lesquelles les piétons, tout comme le fond, restent visuellement proches.

Dans la deuxième phase, toutes ces connaissances sont intégrées dans le détecteur. Pour chaque région, un classifieur est construit à l'aide de la base contextualisée et fonctionne indépendamment des autres. Ainsi chaque classifieur est entraîné avec des données ayant la même apparence que les piétons qu'il devra détecter. Cela simplifie le problème de l'apprentissage et augmente significativement les performances du système.

Mots clés : vidéosurveillance, détection de piétons, apprentissage statistique, apprentissage semi-supervisé, contextualisation.

Abstract

With the rise of videosurveillance systems comes a logical need for automatic and real-time processes to analyze the huge amount of generated data. Among these tools, pedestrian detection algorithms are essential, because in videosurveillance locating people is often the first step leading to more complex behavioral analyses. Classical pedestrian detection approaches are based on machine learning and pattern recognition algorithms. Thus they generally underperform when the pedestrians' appearance observed by a camera tends to differ too much from the one in the generic training dataset.

This thesis studies the concept of the contextualization of such a detector. This consists in introducing scene information into a generic pedestrian detector. The main objective is to adapt it to the most frequent situations and so to improve its overall performances. The key hypothesis made here is that the camera is static, which is common in videosurveillance scenarios.

This work is split into two parts. First a state of the art introduces the architecture of a pedestrian detector and the different algorithms involved in its building. Then the problem of the contextualization is tackled and a series of experiments validates or not the explored leads. The goal is to identify every part of the detector which can benefit from the approach in order to fully contextualize it.

To make the contextualization process easier, our method is completely automatic and is based on semi-supervised learning methods. First of all, data coming from the scene are gathered. We propose different oracles to detect some pedestrians in order to catch their appearance and to form a contextualized training dataset. Then, we analyze the scene geometry, which influences the size and the orientation of the pedestrians and we divide the scene into different regions. In each region, pedestrians as well as background elements share a similar appearance.

In the second step, all this information is used to build the final detector which is composed of several classifiers, one by region. Each classifier independently scans its dedicated piece of image. Thus, it is only trained with a region-specific contextualized dataset, containing less appearance variability than a global one. Consequently, the training stage is easier and the overall detection results on the scene are improved.

Key words: videosurveillance, pedestrian detection, machine learning, semi-supervised learning, contextualization.

Table des matières

1 Introduction **1**
1.1 Cadre de l'étude 1
1.2 Contextualisation d'un détecteur de piétons 6
1.3 Présentation du mémoire et travaux réalisés 8

I État de l'art **11**

2 Détection de piétons **11**
2.1 Introduction 12
2.2 Approches basées sur la discontinuité temporelle 14
2.3 Approches basées sur la reconnaissance de formes 18
2.4 Conclusion 40

3 Contextualisation d'un détecteur **41**
3.1 Introduction 42
3.2 Limites des détecteurs génériques 42
3.3 Évaluation d'un détecteur de piétons 44
3.4 Travaux préliminaires justifiant la contextualisation 51
3.5 Apprentissage pour la contextualisation 53
3.6 Conclusion 63

II Travaux réalisés **65**

Positionnement de la thèse **65**

4 Contextualisation par oracle **69**
4.1 Introduction 70
4.2 Définition et propriétés d'un oracle 70
4.3 Oracle 2D 71
4.4 Oracle 3D 95
4.5 Conclusion 106

5 Stratégies de sélection des exemples issus de l'oracle **109**
5.1 Introduction 110
5.2 Sélection aléatoire 112
5.3 Sélection par score, par apparence et par suivi 117
5.4 Sélection par contraintes géométriques et spatiales 124
5.5 Conclusion 135

6 Application à un système automatique de détection de piétons **137**
6.1 Introduction 138
6.2 Hypothèses de recherche 138
6.3 Intégration de l'oracle dans le détecteur 144
6.4 Réglage automatique du seuil de détection 154
6.5 Conclusion 163

7 Conclusion et Perspectives **167**

Bibliographie **171**

1

Introduction

Ces dernières années, l'augmentation de la puissance de calcul des outils informatiques a permis l'émergence de la vision par ordinateur, dont le but est de permettre à une intelligence artificielle de comprendre une image. Cette technologie est à la base de nombreuses applications comme la robotique, la réalité augmentée...

Dans cette thèse, nous nous intéressons à la vidéosurveillance dite « intelligente », c'est-à-dire à l'étude des algorithmes visant à perfectionner les systèmes traditionnels qui reposent généralement sur la vigilance d'opérateurs humains. Notre but principal est de rendre les détecteurs de piétons plus performants grâce à la prise en compte du contexte. Cela signifie, enrichir automatiquement le détecteur avec des informations caractéristiques de la scène observée.

1.1 Cadre de l'étude

Cette partie expose les motivations de cette étude due à l'importance du développement actuel des technologies de vidéosurveillance et de détection de piétons.

La vidéosurveillance

La vidéosurveillance répond à une demande forte émanant de la société. Elle consiste à déployer, dans un lieu public ou privé, un ensemble de caméras afin d'observer et de surveiller ce qu'il s'y passe. De tels systèmes sont installés en priorité dans les centres-villes, les transports en commun, les parkings, sur le réseau routier ou dans certains sites sensibles (administrations, sites industriels...). Bien que les premières utilisations de la vidéosurveillance datent du milieu du XX^{e} siècle, c'est réellement à partir des années 1990-2000 que la technologie prend son essor.

L'objectif premier d'un système de vidéosurveillance est d'améliorer la sécurité générale des biens et des personnes. Cet outil permet de lutter contre la plupart des types de délinquance commis sur la voie publique et facilite la mission des forces de l'ordre et des magistrats. En ce sens les attentes envers ces dispositifs sont très importantes. Grâce à la vidéo, il est par exemple possible d'observer un lieu, d'identifier ou de suivre des personnes et des objets, d'analyser de manière statistique les flux de déplacement d'une foule ou des véhicules et de détecter des anomalies comme des événements rares (voiture remontant une file en sens inverse). Il est ainsi possible d'espérer une réduction du nombre de délits grâce au pouvoir de dissuasion ou une augmentation du taux d'élucidation des investigations (aide à la filature, identification à distance de personnes recherchées...).

Néanmoins une part importante de l'apport de la vidéosurveillance réside dans son aspect préventif. Dans le cadre de la prévention routière, la détection des conduites dangereuses, l'analyse en temps réel de l'état du trafic, la mise en garde du public sur des dangers potentiels, les systèmes d'aide à la conduite et la lutte contre les infractions au stationnement sont autant d'applications permettant une amélioration des conditions de sécurité. Dans les transports, la surveillance du réseau rendrait possible l'intervention prompte des services de sécurité lors de la découverte d'un colis abandonné ou de l'obstruction de voies de chemin de fer...

Dans un contexte industriel, les caméras permettent le contrôle des accès d'une installation sensible (outils de reconnaissance faciale) et la détection d'éventuelles intrusions. Elles peuvent aussi aider à contrôler le bon déroulement d'un processus et à inspecter l'état des installations en temps réel.

Dans des problématiques d'aide à la personne, les gens égarés dans un hôpital peuvent être repérés plus facilement et plus rapidement afin de les aider. Dans le contexte du maintien à domicile, un système capable d'analyser la posture d'un humain pourrait avertir les secours en cas de chute ou de malaise.

Ces dernières années, de nombreux pays se sont équipés de systèmes de vidéosurveillance [Dee et Velastin, 2008]. Le Royaume-Uni est certainement le pays comptant le plus de caméras par habitant : entre 1,8 million et 4,2 millions de caméras (toutes caméras confondues) pour 63 millions de Britanniques.

En France, le gouvernement a encouragé à partir de 2007, l'installation de systèmes de vidéosurveillance sur la voie publique avec comme objectif de tripler leur nombre. Suivant les sources, mentionnées dans [Migaud *et al.*, 2011], le nombre de caméras installées sur la voie publique serait compris entre 10 000 et 40 000 en 2010.

Il est délicat de dresser un premier bilan de l'exploitation à grande échelle de ces technologies encore immatures, car au final peu d'études sont conduites sur le sujet. L'évaluation de l'efficacité des dispositifs de vidéosurveillance pour la lutte contre la criminalité est difficile. La mesure de la baisse ou non de la délinquance n'est pas un indicateur suffisant, puisque la vidéosurveillance n'est qu'un outil parmi d'autres et doit s'intégrer dans un ensemble plus global. La criminalité est un phénomène complexe pour lequel de nombreux paramètres entrent en ligne de compte : niveau des installations, présence policière, conditions sociales...

Le rapport publié par la Cour des Comptes [Migaud *et al.*, 2011] esquisse un bilan mitigé et critique la mise en place des systèmes actuels. Outre les problèmes liés au respect de la vie privée (espionnage de la population, observation potentielle d'appartements privés par les fenêtres...), se pose la question de l'efficacité et du coût de ces équipements.

Selon certaines sources, l'apport de la vidéosurveillance est encourageant. Dans les espaces clos, dans les parkings ou dans les transports équipés, une diminution importante de la petite criminalité est constatée. Néanmoins, il semblerait que les systèmes déployés à l'échelle d'une ville ont un effet peu significatif. La Cour des Comptes avance dans son rapport, que 3% des affaires élucidées sur la voie publique le sont grâce à la vidéosurveillance.

L'investissement lié à l'installation d'un système de vidéosurveillance dans l'espace public est élevé. Cela nécessite du matériel ainsi que sa mise en place, son raccordement au réseau et un système de traitement... Les chiffres avancés sont très variables d'une commune à l'autre. En moyenne l'installation d'une caméra représente un coût d'environ 20 000 euros tout compris. À cela s'ajoutent les frais d'exploitation (rémunération du personnel, entretien et remplacement du matériel défectueux...) chiffrés en moyenne à 7 400 euros par an et par caméra.

Le financement des outils de vidéosurveillance est aujourd'hui considérable mais ceux-ci restent sous-exploités en raison d'un manque de moyens humains. En effet, l'un des plus gros problèmes posés à la vidéosurveillance est le traitement de la masse de données. Le nombre de caméras est trop important pour que toutes soient visionnées en permanence et les opérateurs sont obligés de passer successivement d'une vue à l'autre. Dans leur analyse, [Dee et Velastin, 2008] estiment que dans le meilleur des cas, le flux d'une caméra sur quatre est observé. Ce rapport peut descendre à 1/78. Ainsi seulement un faible pourcentage de vidéos est visionné. Pire, le travail des agents est extrêmement répétitif et lassant du fait du peu de situations anormales. Cela rend la probabilité d'observer un flagrant délit relativement faible et de nombreux systèmes ne sont utilisés qu'*a posteriori*. Malheureusement, la recherche est difficile car repérer un événement qui n'a duré que quelques secondes parmi des heures de vidéo nécessite un travail lourd et peu gratifiant. Avec les systèmes actuels l'humain est au cœur du processus. Comme il est très difficile pour un agent de maintenir son attention en permanence et de surveiller beaucoup de caméras à la fois, les performances globales des dispositifs sont limitées.

Une piste pour remédier à ces problèmes est le développement de la « vidéosurveillance intelligente ». Son but est d'aider les opérateurs en les avertissant lorsque quelque chose d'anormal se produit, puis à terme de remplacer ces métiers difficiles. De nombreux défis techniques restent à résoudre pour voir apparaître de tels outils pleinement fonctionnels. Une des premières étapes d'un système consiste souvent à détecter la présence d'humains dans la scène. Cette information est importante car elle conditionne bien souvent la suite des traitements.

Dans cette thèse nous nous intéresserons à la détection de piétons dédiée à la vidéosurveillance. Cela implique en particulier un traitement en temps réel des images (au moins 10 images par seconde) provenant d'une caméra fixe.

La détection de piétons

La détection de piétons est une discipline apparue à la fin des années 1990. Elle consiste à déterminer le plus précisément possible, la présence et la localisation de toutes les personnes dans une image. Elle est présente dans de nombreuses applications et est souvent un élément de base d'un traitement plus complexe.

Ses applications ne se cantonnent d'ailleurs pas à la vidéosurveillance. De nombreux systèmes d'aide à la conduite (ou *ADAS* pour *Advanced Driver Assistance Systems*) reposent sur cette brique pour avertir le conducteur en cas de danger. Ce champ de recherche reste encore très actif aujourd'hui du fait de son importance et des difficultés sous-jacentes.

La détection de piétons est une technologie complexe reposant sur de nombreux composants logiciels. Elle est souvent basée sur des approches de reconnaissance de formes dont le but est d'apprendre puis de retrouver dans l'image, l'allure générale d'un humain. La figure 1.1 montre le résultat attendu d'une détection : chaque piéton est entouré par un rectangle englobant.

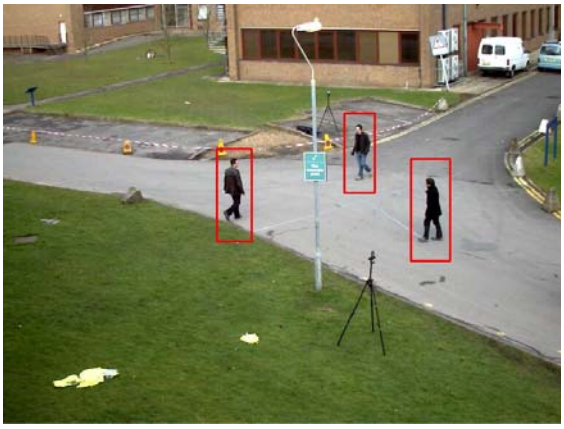


FIGURE 1.1 – Exemple de résultat obtenu à l'aide d'un détecteur de piétons.

Cependant de nombreux facteurs peuvent fortement influer sur la silhouette d'une personne. Le premier paramètre à prendre en compte est la nature même d'un humain. Un piéton est imprévisible. Il est aussi articulé et peut prendre des poses très diverses. Contrairement à une voiture faite d'un seul bloc, l'apparence et la direction d'une personne peuvent changer à tout instant rendant plus difficile sa détection et son suivi. Deuxièmement, les situations dans lesquelles peut se retrouver une personne sont très variées. Par exemple, est-elle seule, à proximité d'un groupe de personnes ou dans une foule ? Est-elle occultée par un objet ? Porte-t-elle des bagages ou tout autre objet modifiant son apparence ? Dans quel type d'environnement évolue-t-elle ? L'arrière-plan est-il simple, dégagé ou au contraire complexe avec de nombreuses structures pouvant ressembler à un humain ? Pour finir, les conditions de prise de vue accentuent encore cette diversité. La position de la caméra, sa hauteur, son orientation, sa résolution mais aussi les conditions extérieures comme la météo ou la présence d'éclairages artificiels sont autant de paramètres susceptibles de changer l'apparence du piéton dans l'image.

Tout ceci complexifie considérablement le problème de la reconnaissance de la forme d'un humain et donc les algorithmes de détection. Cela engendre mécaniquement une hausse des temps de calculs. Pour simplifier les traitements et les rendre compatibles avec le temps réel, il n'est pas rare de se limiter à un sous-ensemble de situations. Un cas classique en vidéosurveillance consiste à prendre uniquement en compte les piétons qui se tiennent debout. Bien sûr cela ne convient pas à toutes les applications et il pourrait être utile dans un établissement de santé d'émettre une alarme si un patient gît à terre.

Une approche possible pour réaliser un module de détection de piétons est de se baser sur des algorithmes de reconnaissance de formes dont le but est de discriminer la forme d'une personne par rapport au fond. Concrètement la première étape consiste à capturer la diversité d'apparence des humains et du fond grâce à la construction d'une base d'apprentissage. Celle-ci doit être la plus grande et la plus variée possible pour rendre compte de la richesse des situations pouvant être rencontrées par le détecteur. La figure 1.2 montre un échantillon d'une base d'apprentissage dédiée à la détection de piétons avec les deux catégories d'images.

(a) Images de fond (b) Images de piétons

FIGURE 1.2 – Échantillon d'une base d'apprentissage dédiée à la détection de piétons contenant des images de fond et de piétons. (source : INRIA Person Dataset)

Dans un deuxième temps, un algorithme d'apprentissage va étudier et dégager des propriétés de cette base pour créer un modèle représentant une frontière entre les deux classes. Un classifieur s'appuie ensuite sur ce modèle pour différencier les éléments des deux groupes.

Une fois le classifieur construit, la phase de détection consiste à découper l'image en rectangles qui se recouvrent, puis à comparer leur contenu avec le modèle. Si en un point donné, l'image est suffisamment proche du modèle de piéton construit précédemment alors le détecteur indique la présence d'un piéton.

1.2 Contextualisation d'un détecteur de piétons

Nous venons de voir que les algorithmes usuels de détection de piétons reposent sur l'utilisation d'une base d'apprentissage pour apprendre les différences entre une personne et le fond.

Intuitivement il est aisé de comprendre qu'avec une telle approche, un détecteur repère beaucoup mieux les événements qu'il a déjà rencontrés auparavant. Pour obtenir des performances maximales, il faut que la base d'apprentissage et les vidéos visionnées partagent des caractéristiques similaires. Or les scènes rencontrées en vidéosurveillance sont très variées (figure 1.3) et il est utopique de croire, en l'état actuel des connaissances, qu'il suffirait de construire une immense base d'apprentissage contenant toutes les situations possibles pour résoudre le problème. La création de bases d'apprentissage de qualité reste une activité manuelle. Capturer la variabilité d'apparence des piétons mais aussi celle des exemples de fond, qui est encore plus vaste, requerrait une masse de travail colossale. D'autre part, cette base d'apprentissage serait extrêmement complexe à appréhender pour un algorithme d'apprentissage. Intuitivement, plus les données sont nombreuses, plus il est difficile de les séparer correctement. Traiter toute cette information nécessiterait alors des moyens de calculs très importants. En l'état actuel des connaissances, il est donc difficile pour un système de prendre en compte tous ces cas de figure afin de fonctionner de manière universelle.

FIGURE 1.3 – Ces deux images comportent plusieurs difficultés pour un détecteur. Certains piétons sont penchés, occultés, observés à des échelles différentes, d'autres transportent des bagages... Le décor est aussi très spécifique avec la présence de mannequins. Une base d'apprentissage générique (figure 1.2) ne contenant que des piétons droits et des éléments de fond quelconques ne peut pas servir à la création d'un détecteur de piétons performant et universel.

Nous avons donc choisi d'explorer une autre voie. Plutôt que de construire un détecteur générique capable de fonctionner dans toutes les situations, nous avons décidé de réaliser un détecteur et de l'enrichir avec des informations provenant de la scène qu'il sera amené à traiter : c'est la contextualisation. L'enjeu est alors d'inclure directement dans le détecteur, les éléments de contexte de la scène dont il a besoin. Cela passe évidemment par

la base d'apprentissage mais aussi par la connaissance de la structure du lieu observé afin d'optimiser la recherche dans l'image. Ainsi, le classifieur a de meilleures performances et il est plus simple car il a été entraîné à reconnaître uniquement les événements qui le concernent.

Dans le cadre de la vidéosurveillance, cette approche soulève de nombreuses difficultés. Il est en effet inconcevable de fabriquer manuellement, comme cela se fait actuellement, une base spécifique pour chaque caméra vu les millions de systèmes déployés dans le monde. Il semble aussi fastidieux de devoir optimiser manuellement tous les paramètres d'un détecteur pour toute nouvelle installation. Cela requerrait du temps et probablement les compétences d'un expert en vision.

Notre approche tente de répondre à ces problèmes en automatisant le travail de contextualisation nécessaire afin de maximiser les performances tout en minimisant la charge de travail des opérateurs humains. L'enjeu de la contextualisation est d'extraire l'information présente dans la scène afin de la transmettre au détecteur et ainsi d'améliorer ses performances. Pour être exploitable dans le cadre de l'installation d'un réseau de caméras important, la contextualisation doit être automatisée.

La procédure de contextualisation, que nous avons développée, est représentée sur la figure 1.4. Le flux vidéo filmant la scène est d'abord analysé. D'une part, l'apparence des piétons et des éléments du fond est extraite et collectée dans une base d'apprentissage dite contextualisée : c'est le rôle de l'oracle. La base ainsi créée est alors utilisée par un algorithme d'apprentissage lors de la formation du détecteur contextualisé. D'autre part les contraintes géométriques s'exerçant sur la scène peuvent renseigner sur la présence potentielle de piétons dans une zone de l'image ou sur leur apparence. Il convient donc de prendre également en compte ces informations dans le détecteur contextualisé.

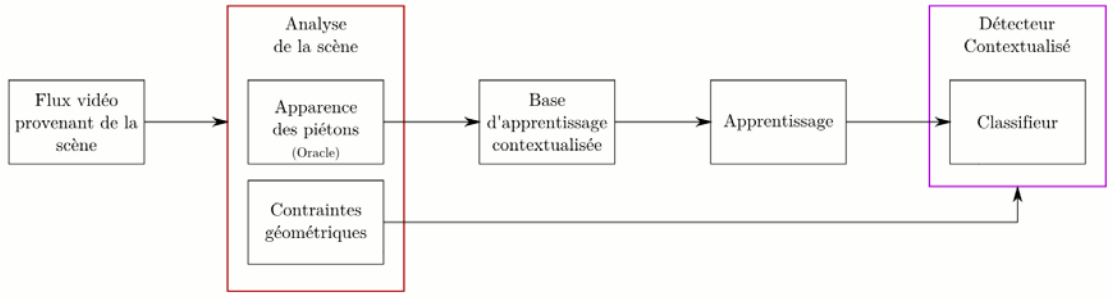


FIGURE 1.4 – Chaîne de contextualisation

1.3 Présentation du mémoire et travaux réalisés

Ce mémoire est découpé en 2 parties. La première comporte deux chapitres et est dédiée à l'état de l'art en vue de justifier les choix effectués sur l'architecture générale du système. La seconde est découpée en trois chapitres et présente les travaux réalisés au cours de la thèse.

Plus précisément, le chapitre 2 s'intéresse à l'état de l'art sur les méthodes de détection de piétons. La littérature dans ce domaine est très riche et de nombreuses variantes existent pour chaque composante du détecteur. Deux grandes familles d'algorithmes s'affrontent. Les premiers, basés sur la discontinuité temporelle du signal vidéo, sont très souples mais repèrent tous les objets en mouvement. À l'inverse les seconds reposent sur la reconnaissance de formes. Ils présentent l'avantage de détecter uniquement les piétons mais sont plus rigides car ils ne s'adaptent pas facilement à un nouveau point de vue.

Le chapitre 3 aborde les enjeux de la contextualisation. Il démarre par la présentation des erreurs les plus fréquemment rencontrées lors de l'utilisation d'un détecteur de piétons générique. Nos travaux préliminaires confirment alors l'intérêt de la contextualisation et montrent très nettement l'apport d'une telle solution. La suite de ce chapitre est donc consacrée à l'étude des différentes approches existantes pour la contextualisation d'un détecteur de piétons.

Après avoir vérifié l'importance de la contextualisation pour un détecteur de piétons, la seconde partie du mémoire commence par présenter notre positionnement par rapport à l'état de l'art et les spécificités de notre approche.

Notre méthode de contextualisation repose sur l'utilisation d'un oracle dont le rôle est de collecter des exemples de piétons et de fond en provenance de la scène. Afin que l'apprentissage contextualisé trouve une bonne séparation entre les deux classes, il est primordial que l'oracle commette le moins d'erreurs possible. L'oracle emploie donc une combinaison d'algorithmes de détection de piétons : reconnaissance de formes afin de garantir que la base ne contient que des piétons et discontinuité temporelle pour valider les observations. Le chapitre 4 décrit les différents oracles proposés : un oracle 2D capable de travailler sans *a priori* sur la scène et un oracle 3D utilisable lorsque les paramètres de la caméra et la géométrie de la scène sont connus.

Le chapitre 5 expose plusieurs stratégies de sélection des exemples contenus dans la base fournie par l'oracle. Une fois que l'oracle a extrait les exemples de la scène, comment organiser la base pour obtenir le meilleur détecteur possible ? Dans cette optique, nous proposons la création d'un « classifieur par régions » pour prendre en compte la structure de la scène. Ceci enrichit le modèle global de représentation des piétons qui est un des facteurs limitant le système. Par contre, localement, la structure du modèle n'est pas modifiée. Au final, les performances du détecteur sont améliorées sans pour autant augmenter les temps de calculs lors de la phase de détection.

Jusqu'à présent, seule la manière de construire un classifieur contextualisé a été abordée. Or un détecteur de piétons est composé de nombreuses briques qui peuvent elles

aussi être adaptées à la scène. Nous nous intéressons donc dans le chapitre 6 à d'autres aspects d'un détecteur. La phase de détection consiste à balayer l'image à la recherche de piétons. Cette étape est coûteuse en temps de calculs et critique pour les systèmes devant fonctionner en temps réel. Pour ne pas gaspiller de ressources, nous proposons de définir les zones de l'image pouvant contenir des piétons et de balayer uniquement ces dernières. Enfin en fonction de ses paramètres et notamment de son seuil de détection, le système peut être plus ou moins sensible. Ce seuil doit être réglé avec précision. S'il est trop bas de nombreuses fausses alarmes seront présentes et s'il est trop élevé les piétons ne seront pas correctement détectés. Une méthode automatique, spécifique à la contextualisation, est présentée pour estimer le seuil du détecteur. Toutes ces étapes sont indispensables pour la mise en œuvre d'un système complet et automatique de contextualisation.

Le mémoire est clos par un chapitre de conclusion et de perspectives.

Liste des publications

Les travaux réalisés dans le cadre de cette thèse ont fait l'objet de plusieurs publications mentionnées ci-dessous.

- [Chesnais *et al.*, 2013] T. Chesnais, T. Chateau, N. Allezard, Y. Dhome, B. Meden, M. Tamaazousti et A. Chan-Hon-Tong : *A Region Driven and Contextualized Pedestrian Detector.* In VISAPP, 2013.
- [Chesnais *et al.*, 2012a] T. Chesnais, N. Allezard, Y. Dhome et T. Chateau : *Automatic Process to Build a Contextualized Detector.* In VISAPP, 2012a.
- [Chesnais *et al.*, 2012b] T. Chesnais, N. Allezard, Y. Dhome et T. Chateau : *Création automatique d'un détecteur adapté à la scène.* In RFIA, 2012b.
- [Dhome *et al.*, 2012] Y. Dhome, B. Luvison, T. Chesnais, R. Belaroussi, L. Lucat, M. Chaouch et P. Sayd : *Outils d'analyse vidéo : Pour une pleine exploitation des données de vidéoprotection*, Chapitre : *Détection d'objets d'intérêt.* Hermès Science Publications, 2012.

Partie I : État de l'art

2

Détection de piétons

Sommaire

2.1 Introduction 12
2.2 Approches basées sur la discontinuité temporelle 14
2.2.1 Soustraction de fond 14
2.2.1.1 Principe 14
2.2.1.2 Applications 16
2.2.2 Flot optique 16
2.2.3 Conclusion 18
2.3 Approches basées sur la reconnaissance de formes 18
2.3.1 Descripteur 19
2.3.1.1 Les ondelettes de Haar 20
2.3.1.2 Descripteurs binaires 21
2.3.1.3 Matrice de covariance 22
2.3.1.4 Histogrammes de gradients orientés 23
2.3.1.5 Choix du descripteur dans nos travaux 24
2.3.2 Apprentissage supervisé 24
2.3.2.1 Introduction à l'apprentissage statistique 24
2.3.2.2 Principe de l'apprentissage supervisé 26
2.3.2.3 Choix de l'algorithme d'apprentissage dans nos travaux 32
2.3.3 Mise en œuvre d'un détecteur 32
2.3.3.1 Modèle de représentation des piétons 33
2.3.3.2 Structure du classifieur 33
2.3.3.3 Stratégie d'exploration du détecteur 35
2.3.3.4 Post-traitements : regroupement des détections 37
2.3.3.5 Choix pour la mise en œuvre du détecteur dans nos travaux 39
2.4 Conclusion 40

2.1 Introduction

Dans ce chapitre sont présentées les différentes méthodes actuellement utilisées en vidéosurveillance pour construire un système de détection de piétons. Deux approches complémentaires existent.

La première consiste à identifier les parties de l'image qui ne sont pas fixes en étudiant les discontinuités temporelles présentes dans le signal vidéo (partie 2.2, page 14). Le résultat est une segmentation de l'image représentant d'un côté le fond et de l'autre les objets mobiles. Ces algorithmes travaillent souvent directement au niveau des pixels, par conséquent la sémantique de leur réponse est assez pauvre. Il n'y a par exemple aucune information sur la nature de l'objet détecté qui peut d'ailleurs n'être segmenté que partiellement. Dans les paragraphes 2.2.1 et 2.2.2 nous expliquerons brièvement les enjeux des méthodes de soustraction de fond (parfois appelées segmentation fond/forme) et de flot optique qui sont les principaux représentants de cette famille d'outils.

La deuxième approche se base sur la reconnaissance de formes (partie 2.3, page 18). Il s'agit d'apprendre l'apparence d'un type d'objets tel un piéton et de retrouver cette forme directement dans les images. La sémantique de la réponse est plus riche puisque le détecteur est programmé pour ne repérer qu'un seul type d'objets. De nombreux algorithmes ont été développés pour fabriquer des détecteurs performants qui se basent pour la plupart sur des outils de classification.

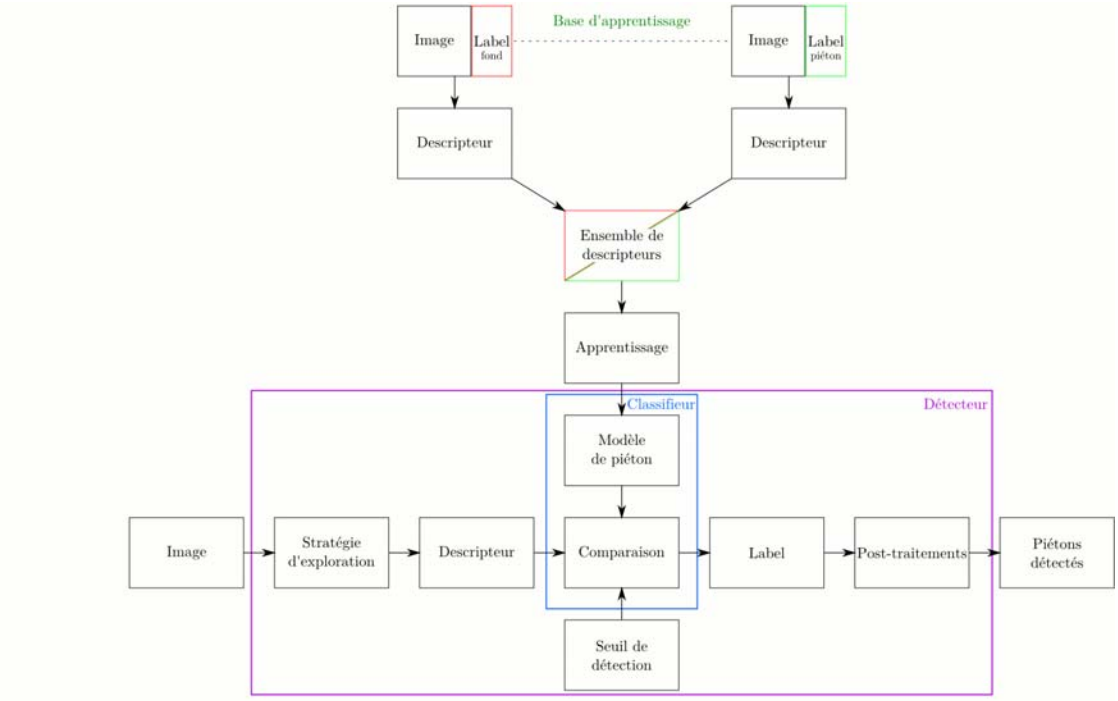


FIGURE 2.1 – Synoptique générique d'un détecteur de piétons basé sur la reconnaissance de formes.

Un système de détection d'objets et *a fortiori* de piétons, basé sur la reconnaissance de formes, est un système modulaire, dont les différents blocs sont indiqués sur la figure 2.1. Deux étapes principales se dégagent : la création d'un modèle (représentée verticalement) et la recherche des objets dans une image (représentée horizontalement).

Les différents blocs du diagramme sont maintenant présentés succinctement. Dans la partie 2.3 de ce chapitre, nous reviendrons plus en détails sur l'architecture d'un détecteur de piéton standard : les descripteurs (paragraphe 2.3.1), les méthodes d'apprentissage (paragraphe 2.3.2) et la mise en œuvre du détecteur proprement dit (paragraphe 2.3.3).

Base d'apprentissage : L'objectif des approches par reconnaissance de formes est de trouver dans une image les endroits contenant un piéton et donc de séparer les piétons du fond. Pour cela, un modèle s'appuyant sur l'apparence est construit pour différencier les piétons du fond . Le rôle de la base d'apprentissage (ou ensemble d'apprentissage) est de collectionner des éléments des deux classes : piétons et fond. Elle contient donc un ensemble d'images labellisées appartenant aux deux classes. Par extension, il est possible d'appelé base d'apprentissage l'ensemble des descripteurs calculés à partir des images étiquetées.

Descripteur : Une image numérique est une matrice. Chaque nombre y code l'intensité lumineuse d'un pixel. Pour représenter une image en couleurs, il faut une matrice par canal (rouge, vert, bleu). Une vidéo est une succession d'images et donc une succession de matrices. Ces grandes quantités de données brutes ne permettent pas de détecter aisément les objets puisque les liens spatiaux (mais aussi temporels dans le cas d'une vidéo) entre les différents pixels ne sont pas visibles. Le rôle du descripteur est d'extraire à partir des données brutes de l'image, des informations exploitables par le système.

Il existe plusieurs types de descripteurs. Le choix est conditionné par l'application visée puisque c'est elle qui détermine les informations utiles. Par exemple, pour repérer des modifications entre des images proches, la couleur est très importante tandis que pour reconnaître la forme d'une personne un descripteur basé sur les contours est plus approprié. Généralement, le résultat du calcul d'un descripteur est un vecteur. Le passage de l'image numérique au descripteur est réalisé par une primitive de descripteur.

Par abus de langage, le terme descripteur sert à désigner suivant les cas le type de descripteur, le vecteur descripteur ou la primitive de descripteur.

Apprentissage : La base de descripteurs est ensuite traitée par un algorithme d'apprentissage qui va en extraire un **modèle**. Ce dernier peut être de nature différente selon l'algorithme employé. Généralement, le modèle représente les contours importants de l'objet recherché qui sont discriminants par rapport aux éléments du fond.

Comparaison : Il s'agit ici de mesurer la similarité du modèle avec un descripteur provenant de l'image étudiée (ou image courante). S'ils sont suffisamment proches, le système considère qu'il s'agit d'un objet du type recherché. Le système peut être plus ou moins sensible en fonction du réglage du **seuil de détection**. Plus le seuil est haut, moins le détecteur commettra d'erreurs mais plus la probabilité de ne pas déceler la présence des piétons est grande.

Classifieur : Le classifieur est une fonction qui à un descripteur associe son label (piéton ou fond). Il est constitué par le modèle appris par l'algorithme d'apprentissage et le module de comparaison.

Stratégie d'exploration : Une fois le modèle établi, il doit être retrouvé dans une image. Il est donc important de définir une stratégie de parcours du détecteur. Il est possible d'explorer toute l'image ou de se limiter à un voisinage de la dernière position connue d'un objet qui a déjà été identifié dans les images précédentes.

Post-traitements : Souvent la comparaison avec le modèle est trop peu sélective et la sortie du détecteur est bruitée autour des objets. Parfois le modèle ne représente pas directement l'objet recherché mais une partie seulement et plusieurs modèles sont utilisés en parallèle pour assurer la détection. Pour pallier ces problèmes, une étape de post-traitements sert à fusionner ou filtrer les résultats.

Détecteur : Le détecteur est constitué par l'ensemble des briques nécessaires lors de la phase de détection (stratégie d'exploration, classifieur, post-traitements...).

2.2 Approches basées sur la discontinuité temporelle

De nombreuses voies ont été explorées pour détecter des piétons. Les premières méthodes étudiées dans les caméras stationnaires sont basées sur la segmentation de l'image en fond et en objets en mouvement [Stauffer et Grimson, 2000; Zhao et Nevatia, 2003].

Ces méthodes travaillent généralement de manière locale. Elles labellisent chaque pixel indépendamment des autres uniquement grâce à l'information qu'il contient. Il n'y a donc aucune notion d'objets dans leur réponse. Une étape de post-traitement est alors souvent nécessaire pour donner du sens à la segmentation brute.

2.2.1 Soustraction de fond

2.2.1.1 Principe

La soustraction de fond est une méthode permettant de séparer les objets mobiles du fond de l'image. Elle est basée sur l'idée que le fond est statique. Si une portion de l'image ne change pas pendant un certain intervalle de temps alors elle est considérée comme faisant partie du fond, dans le cas contraire il s'agit d'un objet. Même si des extensions aux caméras mobiles ont été développées, la plupart des approches supposent une caméra fixe.

De nombreuses difficultés techniques existent et sont bien résumées par [Toyama *et al.*, 1999] : variation de la luminosité, ombres portées des objets, mouvement de la caméra (vibrations), signal vidéo de mauvaise qualité, bruité ou saturé. Enfin il n'est pas rare qu'un pixel du fond puisse prendre successivement plusieurs couleurs. Par exemple, l'observation du feuillage d'un arbre peut conduire un pixel à être bleu (ciel), vert (feuille) ou brun (branche). Il est préférable de ne pas tenir compte de ces mouvements car après tout un arbre est considéré comme immobile et ne devrait pas être détecté. Un tel fond est qualifié de multimodal.

À cause de tous ces problèmes, il n'est pas très efficace de simplement faire la différence entre deux images proches. L'option choisie est alors d'élargir la fenêtre temporelle pour détecter les continuités et les ruptures dans le signal vidéo.

La majorité des algorithmes de soustraction de fond présente une architecture commune :

- Apprentissage d'un modèle du fond,
- Classification des pixels en fond ou en objet suivant leur conformité au modèle,
- Mise-à-jour du modèle du fond.

La méthode de soustraction de fond la plus connue est certainement celle proposée par [Stauffer et Grimson, 1999] car de nombreux travaux s'en sont inspirés par la suite. Concrètement, elle fonctionne de la manière suivante.

L'information pertinente pour un algorithme de soustraction de fond est la couleur. Un descripteur constitué de vecteurs RGB est donc bien adapté. Des améliorations de l'algorithme initial ont employé d'autres formes de descripteurs. Les travaux de [Chen *et al.*, 2007] s'appuient sur un descripteur par bloc qui code les informations de couleurs et de contraste.

Après avoir sélectionné un descripteur, ce dernier est utilisé pour générer un modèle du fond par pixel. Les données d'apprentissage sont d'abord collectées sur plusieurs images (environ 10 à 100). L'hypothèse pour construire un modèle est de considérer que les éléments les plus présents statistiquement dans la base sont les plus représentatifs du fond. Pour chaque pixel, [Stauffer et Grimson, 1999] choisissent de représenter le fond par un mélange de trois ou cinq gaussiennes multidimensionnelles (une dimension par couleur). Une gaussienne permet de modéliser la moyenne et la variance de la couleur d'un pixel. Le mélange sert à traiter les fonds multimodaux. Chaque gaussienne est pondérée en fonction de la probabilité qu'a un pixel d'appartenir à la couleur qu'elle représente.

Une fois le modèle appris, il faut classer tous les pixels provenant d'une nouvelle image en fond ou en objet, ce qui revient à comparer leur valeur avec le mélange de gaussiennes. Les gaussiennes sont ordonnées en fonction du rapport entre leur variance et leur poids. Un poids élevé et une variance faible signifient que la gaussienne représente une couleur précise qui a été aperçue souvent et qu'elle est donc très pertinente. Il est considéré que, si une gaussienne suffisamment pertinente, c'est-à-dire d'un poids supérieur à un certain seuil, peut expliquer la valeur courante d'un pixel, alors il s'agit du fond. La moyenne et la variance de la gaussienne sont alors mises à jour en y intégrant cette valeur. Dans le cas contraire, le pixel est labellisé en tant qu'objet. Il est possible que cet étiquetage résulte d'un changement du fond (une valise immobile sur le sol, une voiture garée, le feuillage d'un arbre qui se déplace avec le vent) qui doit donc être appris. Pour ce faire, la gaussienne la moins pertinente est remplacée par une autre centrée sur la couleur de ce pixel avec un poids faible et une variance élevée.

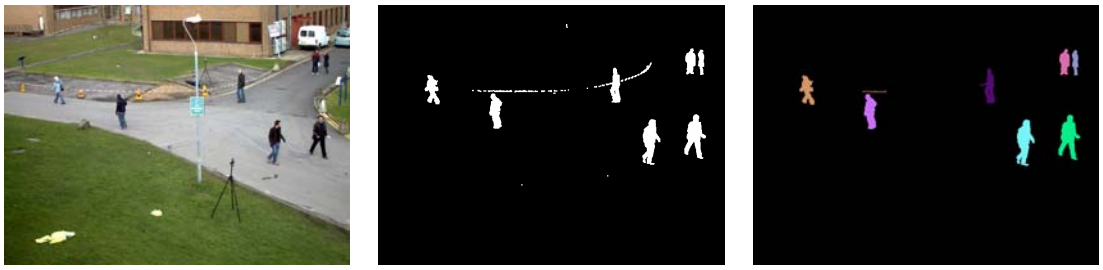
Il existe beaucoup d'autres algorithmes de soustraction de fond. Par exemple, [Kim *et al.*, 2004] utilisent un *codebook*, c'est-à-dire une table stockant pour chaque pixel un échantillon des valeurs passées, formant ainsi un modèle de fond. Les lecteurs intéressés pourront se référer aux travaux de [Piccardi, 2004; Elhabian *et al.*, 2008] qui présentent un état de l'art du domaine.

Ces méthodes sont utiles car si les objets de la scène sont majoritairement des piétons et que ceux-ci se déplacent, elles permettent de les détecter quelle que soit leur apparence. C'est une technique de base en vidéosurveillance. Elle constitue souvent la première étape de la chaîne de traitement car c'est une technologie qui fournit des résultats globalement bons tout en étant relativement légère en temps de calcul.

2.2.1.2 Applications

Les applications de la soustraction de fond sont très nombreuses. Deux sont abordées dans ce paragraphe. Cet outil est rarement utilisé seul, puisque l'information fournie est assez pauvre dans le sens où la notion d'objet n'existe pas. Des algorithmes complémentaires sont nécessaires pour donner du sens à une segmentation binaire de l'image.

Pour réaliser de la détection par soustraction de fond, il est intéressant de définir des objets et de trouver leurs contours. Visuellement, un objet correspond à une « tache » dans la segmentation. L'algorithme proposé par [Chang *et al.*, 2004] est capable d'identifier chaque objet. La figure 2.2 montre la sortie de cet algorithme.



(a) Image originale (PETS 2009) (b) Résultat de la soustraction de fond : segmentation binaire (c) Détections par soustraction de fond. Les « taches » sont colorées pour identifier chaque objet mobile.

FIGURE 2.2 – Utilisation de la soustraction de fond pour détecter les piétons

La soustraction de fond est aussi très souvent utilisée en tracking (suivi d'objets : piétons, véhicules...). Une des difficultés rencontrées par ces algorithmes est l'initialisation des pistes : comment savoir si ce qui est observé est un objet nouveau et intéressant ? La soustraction de fond répond partiellement au problème. Une piste est créée lors de l'apparition d'une nouvelle « tache ». Suivre l'objet, consiste alors à apparier cette « tache » avec celles déjà observées dans des images proches [Stauffer et Grimson, 2000].

2.2.2 Flot optique

Les algorithmes de flot optique ont pour but de caractériser le mouvement dans une scène. À chaque pixel ou bloc est associé un vecteur de déplacement v entre une image source et une image cible.

Ce paragraphe a pour but de décrire brièvement les enjeux de l'estimation du flot optique. Les lecteurs intéressés pourront se référer aux travaux de [Luvison, 2010] pour une description plus détaillée de ces méthodes. La figure 2.3 présente les résultats obtenus grâce à un algorithme de flot optique. La couleur des pixels indique l'angle (bleu : pixels

qui se déplacent vers la gauche, rouge : pixels qui se déplacent vers la droite), tandis que la luminosité est fonction de l'amplitude du mouvement.

Le principe des méthodes de flot optique est le suivant. Entre deux images consécutives, la luminance I d'un objet doit rester la même. Cela revient à estimer v tel que :

$$I(x,t) - I(x + vdt, t + dt) = 0 \tag{2.1}$$

Ce problème appelé « contrainte de conservation des données » est difficile à résoudre car les hypothèses pour le résoudre sont faibles. Tout d'abord comme pour la soustraction de fond, la notion d'objet n'existe pas. Chaque pixel peut donc en théorie bouger indépendamment de ses voisins. D'autre part la luminance n'est pas forcément conservée entre deux images. Un objet peut être occulté, la luminosité d'une scène peut changer... Pour faciliter la résolution, les algorithmes ajoutent l'hypothèse que les pixels proches appartiennent au même objet et doivent donc se déplacer de manière similaire. Cette hypothèse s'appelle la cohérence spatiale.

(a) Image 1 (PETS 2009) (b) Image 2 (c) Flot optique résultant

FIGURE 2.3 – Utilisation du flot optique

Il existe une multitude d'algorithmes de flot optique, qui diffèrent principalement par la formulation des hypothèses exposées précédemment. Selon la classification de [Barron *et al.*, 1994], trois grandes familles d'algorithmes existent.

Les **méthodes différentielles** expriment le problème de conservation à l'aide d'un développement limité.

$$I(x,t) - I(x - vt, 0) = 0 \tag{2.2}$$

À l'aide d'un développement de Taylor du premier ordre par rapport au temps, cette équation devient :

$$\frac{d}{dt}I(x,t) + \frac{d}{dx}I(x,t).v = 0 \tag{2.3}$$

Parmi ces méthodes nous pouvons citer celles de [Lucas et Kanade, 1981; Horn et Schunck, 1981] qui se différencient par la façon d'exprimer la contrainte spatiale.

Les **méthodes par corrélation** font partie de la famille des algorithmes de *Block Matching*. Elles visent à mettre en correspondance une fenêtre de taille D dans les images source et cible. Pour cela un critère de similarité et une stratégie de recherche sont utilisés.

Les **méthodes par régression** supposent que la loi qui régit le déplacement est connue mais pas ses paramètres. Par exemple [Black, 1996] suppose que le champ de déplacement est affine par morceaux. Le problème de conservation des données est ensuite résolu à l'aide de sa forme différentielle.

Lors de nos expérimentations, notre choix s'est porté sur l'algorithme de [Black, 1996]. Cette méthode fournit une bonne estimation du déplacement et est relativement robuste au bruit.

2.2.3 Conclusion

Les approches basées sur la soustraction de fond ou sur le flot optique sont sensibles à la même information : les discontinuités temporelles dans le signal colorimétrique de la vidéo. La différence majeure entre les deux provient de leur temps de réponse. Le flot optique estime le mouvement entre deux images proches tandis que la soustraction de fond construit un modèle du fond sur quelques dizaines voire centaines d'images qu'elle compare ensuite à l'image courante. Elle a donc un temps d'intégration plus long.

Ces méthodes détectent bien les objets en mouvement quels qu'ils soient. Comme nous le verrons par la suite, elles sont plus souples que les approches de reconnaissance de formes car elles peuvent être utilisées dans n'importe quel type d'environnement. Malheureusement elles possèdent quelques défauts. Elles restent sensibles au bruit et aux variations de l'environnement mais surtout leurs réponses manquent de sémantique. Il n'y a aucun moyen fiable pour savoir si un objet a été correctement segmenté ou pour déterminer la nature de celui-ci puisque tous les objets mobiles sont repérés sans distinction. Or notre but est de détecter les piétons et uniquement eux. Dans ce cas, ces algorithmes ne sont pas suffisants et viennent en complément d'une autre approche : la reconnaissance de formes.

2.3 Approches basées sur la reconnaissance de formes

Les approches de détection d'objets basées sur la reconnaissance de formes utilisent des algorithmes d'apprentissage statistique pour séparer le fond des objets recherchés. Ces méthodes permettent de reconnaître l'apparence ou l'allure générale des objets, ici des piétons. Pour ce faire elles fabriquent un modèle discriminant plus ou moins riche de ce qui constitue un piéton. L'avantage évident par rapport aux méthodes précédentes est qu'elles ne sont sensées détecter que les piétons et non tous les objets en mouvement. Dans cette partie nous nous intéressons plus particulièrement aux méthodes de détection de piétons compatibles aujourd'hui avec le temps réel (moins de 100 millisecondes pour traiter une image).

La reconnaissance de formes fait souvent appel aux techniques de classification. Cette thèse s'intéresse exclusivement à la classification binaire qui comme son nom l'indique permet de différencier deux classes, à savoir les piétons contre tout le reste. Ces dernières années la classification a été très utilisée en vision par ordinateur. Parmi les travaux

précurseurs, nous pouvons noter ceux de [Papageorgiou *et al.*, 1998], étendus ensuite par [Viola et Jones, 2001b; Viola *et al.*, 2005]. Des articles récents présentent et comparent les dernières avancées dans le domaine [Dollár *et al.*, 2011, 2010; Enzweiler et Gavrila, 2009; Gerónimo *et al.*, 2010].

Dans la suite de ce chapitre, les principaux concepts utilisés dans la détection de piétons, mentionnés sur la figure 2.1 (page 12), seront détaillés. Le but n'est pas de présenter tous les aspects techniques des différents algorithmes de manière précise mais de passer en revue les éléments les plus importants pour donner une idée au lecteur des méthodes employées. Nous commencerons par étudier quelques descripteurs parmi les plus populaires en détection de piétons (paragraphe 2.3.1, page 19). Puis nous verrons comment traiter toutes ces informations extraites des images à l'aide de la classification (paragraphes 2.3.2, page 24) pour calculer un classifieur. Enfin nous nous intéresserons plus en détails aux différentes manières d'utiliser un détecteur (paragraphe 2.3.3, page 32), aux différentes stratégies de parcours d'une image ainsi qu'aux post-traitements nécessaires pour affiner les sorties brutes d'un classifieur.

2.3.1 Descripteur

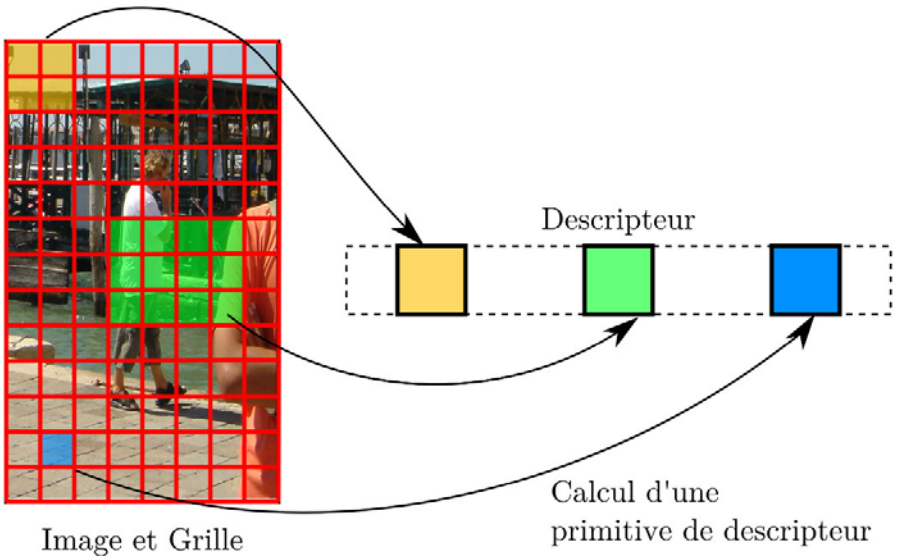


FIGURE 2.4 – Principe de calcul d'un descripteur. Une primitive de descripteur est appliquée sur une grille. Ces primitives sont ensuite concaténées pour former le vecteur descripteur.

L'information très riche contenue dans les images doit d'abord être transformée en un descripteur qui est plus facilement exploitable par les algorithmes de vision par ordinateur. En général, un descripteur, utilisé en détection de piétons, code la localisation spatiale, l'amplitude et l'allure des différences d'intensité dans l'image. Ces informations sont intéressantes car contrairement aux données brutes de l'image, elles permettent, par exemple, de retrouver la tête ou le V inversé formé par les jambes d'une personne. L'enjeu ici étant de détecter tous les piétons et non un en particulier, la couleur ne semble pas

a priori un élément discriminant suffisant et est donc peu utilisée. Une personne peut en effet porter des vêtements de n'importe quelle couleur.

Le descripteur est un organe essentiel du détecteur qui influe grandement sur ses performances. Généralement choisir tel ou tel descripteur revient à faire un compromis entre la richesse de la représentation, le temps de calcul et la taille mémoire nécessaire. En effet, la représentation doit être la plus riche possible. Si trop d'informations sont perdues à ce stade, si les descripteurs de piétons ne peuvent plus être distingués de ceux du fond alors le système ne peut pas différencier les deux classes. Ainsi même en améliorant les autres parties du détecteur, les performances ne seront pas meilleures car le système sature. D'un autre côté, le descripteur doit être le plus compact possible. Dans certaines applications, il est nécessaire de stocker de nombreux descripteurs, par exemple lors d'un apprentissage offline (*cf* page 25). Moins le descripteur occupe de mémoire, plus il est possible d'en stocker et généralement meilleures sont les performances. Enfin le descripteur doit être rapide à calculer car lors de la détection de nombreux descripteurs doivent être évalués.

Le calcul du descripteur, sur une portion de l'image, repose généralement sur le même formalisme. Celui-ci est détaillé sur la figure 2.4. Tout d'abord, la zone de l'image est découpée, à l'aide d'une grille régulière, puis des blocs de tailles et de formes différentes sont construits avec les cases de cette grille. Souvent les blocs sont densément répartis, de forme rectangulaire avec un fort taux de recouvrement entre eux. Chaque bloc est alors encodé à l'aide de la primitive du descripteur précédemment choisie. Une même zone peut donc être codée plusieurs fois. À la fin, les informations de chaque bloc sont concaténées pour former le descripteur (parfois appelé vecteur descripteur).

À noter que généralement, le descripteur est calculé sur une portion de l'image dont les dimensions sont directement reliées à la taille des piétons à détecter et plus importante que le piéton lui-même (figure 2.4). Dans leurs travaux, [Dalal et Triggs, 2005] ont démontré que prendre en compte un peu de fond lors de ce calcul permettait d'améliorer les résultats. Ces marges sont présentes sur tous les bords de l'imagette. Elles font environ 25% de chaque côté horizontalement et 12% verticalement. C'est la raison pour laquelle les détecteurs de piétons fournissent dans la plupart des cas une boîte englobante assez large autour des détections.

De nombreux descripteurs différents ont été proposés dans la littérature. Dans cette présentation nous nous limitons aux principaux qui ont été employés avec succès dans le domaine de la détection de piétons et nous n'abordons dans les calculs que le cas des images en noir et blanc. Le choix du descripteur est difficile et est souvent fait de manière empirique.

2.3.1.1 Les ondelettes de Haar

Les ondelettes de Haar ont été parmi les premiers descripteurs à être employés en détection d'objets [Oren *et al.*, 1997; Papageorgiou *et al.*, 1998; Papageorgiou et Poggio, 2000; Viola *et al.*, 2005].

Une ondelette extrait la différence d'intensité entre deux zones de l'image. En combinant plusieurs ondelettes de tailles et de formes différentes appliquées en de nombreux blocs de l'image, il est possible d'obtenir une représentation de celle-ci.

Concrètement, le calcul d'une ondelette de Haar est très simple et rapide (moyennant l'usage des images intégrales dont le fonctionnement est expliqué page suivante). Il fait

appel à des masques dont les plus courants sont représentés sur la figure 2.5. Une fois le masque de l'ondelette placé aux bonnes dimensions sur les blocs du descripteur, il consiste à faire la différence entre la valeur des pixels sous la partie blanche et ceux sous la partie noire du masque. Par exemple l'évaluation de l'ondelette présentée à gauche sur la figure 2.5, consiste à sommer la valeur des pixels de la partie blanche, à droite, puis à en soustraire la somme des valeurs des pixels de la partie noire, à gauche.

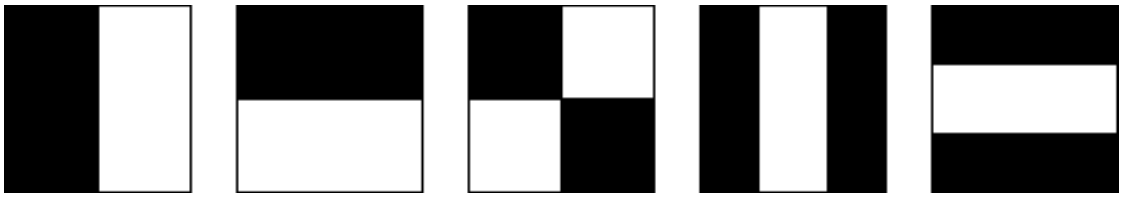


FIGURE 2.5 – Différentes ondelettes de Haar

En fonction de la taille de l'ondelette, sommer la valeur des pixels peut prendre du temps. De plus, lors de l'étape de détection il n'est pas rare de traiter plusieurs fois les mêmes pixels. Il est alors possible d'optimiser le calcul grâce aux images intégrales [Crow, 1984; Viola et Jones, 2001b].

Une image intégrale, $I_{intégrale}$, est une représentation d'une image, I, qui permet un calcul très rapide de la somme des valeurs des pixels contenus dans un rectangle quelconque dont les côtés sont parallèles à ceux de l'image d'origine. Pour chaque image fournie par la caméra, il faut d'abord créer l'image intégrale correspondante. Dans celle-ci, la valeur d'un pixel de coordonnées (x, y) est égale à la somme des valeurs des pixels contenus dans un rectangle englobant le point en haut à gauche de l'image jusqu'au point (x, y) (voir l'équation 2.4 et la figure 2.6(a)).

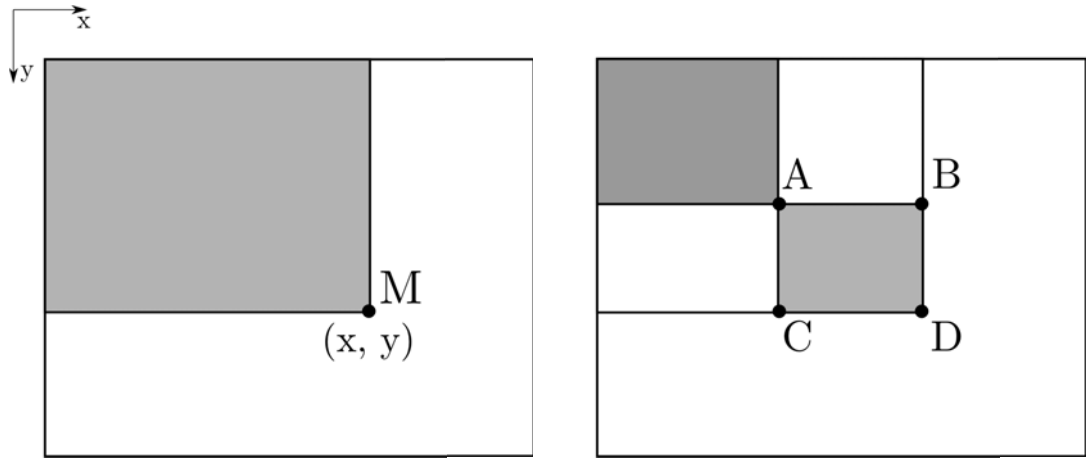
$$M = I_{intégrale}(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \tag{2.4}$$

Pour calculer la somme des pixels du rectangle ABDC (figure 2.6(b)), il suffit d'appliquer la formule suivante $ABDC = A + D - B - C$. Comme annoncé précédemment, cette représentation est très utile pour calculer rapidement la somme des valeurs des pixels d'un rectangle, puisque cela ne nécessite que quatre additions quelle que soit la taille du rectangle. Un détecteur de piétons devra effectuer de nombreuses fois ce type de calcul sur des rectangles qui se recouvrent. L'utilisation des images intégrales est donc particulièrement indiquée.

À noter que d'autres formes et orientations d'ondelettes de Haar ont été proposées. Parmi les travaux les exploitant, il est possible de citer ceux de [Lienhart et Maydt, 2002; Pham *et al.*, 2010].

2.3.1.2 Descripteurs binaires

Comme son nom l'indique un descripteur binaire est un vecteur composé de 0 et de 1. L'intérêt de ce type de représentation est sa compacité et sa rapidité de traitement par les processeurs actuels car il ne nécessite pas de gestion de nombres décimaux. Le représentant



(a) Calcul d'une image intégrale

(b) Utilisation d'une image intégrale pour calculer la somme des pixels d'un rectangle

FIGURE 2.6 – Principe des images intégrales

le plus connu de cette famille est sûrement le local binary pattern ou LBP [Ojala *et al.*, 1996], [Ojala *et al.*, 2002], [Wang *et al.*, 2009]. Ce descripteur est basé sur la différence des niveaux de gris entre un pixel et son voisinage. Le principe de fonctionnement est détaillé sur la figure 2.7. Pour calculer la valeur du descripteur au point C, il faut commencer par tracer un cercle de centre C et de rayon R fixé. N points sont ensuite équirépartis sur le cercle. Seuls les pixels dans lesquels un point existe sont considérés. La valeur du descripteur, en base 2, est :

$$LBP_{C,R,N} = \sum_{i=0}^{N-1} [1 + sgn(I(i) - I(C))]2^{i-1} \tag{2.5}$$

avec I(i) l'intensité du pixel correspondant au point i et sgn(x) le signe de x (-1 ou +1).

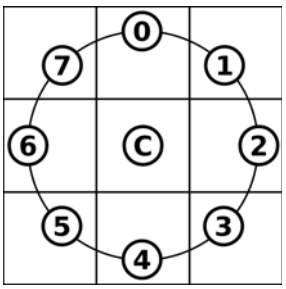


FIGURE 2.7 – Principe du local binary pattern. La différence d'intensité de la luminance est calculée entre le pixel central et ses voisins.

2.3.1.3 Matrice de covariance

Le principe des matrices de covariance [Tuzel *et al.*, 2006] est de coder la moyenne et les variations de certaines caractéristiques de l'image. Pour chaque pixel, il s'agit de

calculer un vecteur de caractéristiques $f(x, y)$. Dans la version originale, les informations de couleurs R, G, B, de forme (gradient de l'intensité I) sont codées :

$$f(x,y) = (x, y, R(x,y), G(x,y), B(x,y), ||\frac{\partial I}{\partial x}||, ||\frac{\partial I}{\partial y}||, ||\frac{\partial^2 I}{\partial x^2}||, ||\frac{\partial^2 I}{\partial y^2}||) \tag{2.6}$$

Une fois les caractéristiques extraites pour chaque pixel, il s'agit de calculer la matrice de covariance de tous les vecteurs d'une portion rectangulaire de l'image.

$$C_R = \frac{1}{n-1} \sum_{x,y} (f(x,y) - \mu)(f(x,y) - \mu)^T \tag{2.7}$$

avec μ la moyenne des vecteurs dans la zone du calcul, n le nombre de pixels dans cette zone.

Les matrices de covariance sont très utilisées en vidéosurveillance. Par exemple dans [Yao et Odobez, 2008], les auteurs ajoutent la valeur de la soustraction de fond dans les caractéristiques des pixels. Elles servent principalement à fusionner différentes caractéristiques qui peuvent être corrélées. Comme pour les ondelettes de Haar, l'utilisation des images intégrales permet d'accélérer les temps de calcul [Tuzel *et al.*, 2008].

2.3.1.4 Histogrammes de gradients orientés

Les histogrammes de gradients orientés (*Histograms of Oriented Gradients* ou HOG) et leurs variantes sont sans conteste les descripteurs les plus populaires dans le domaine de la détection de piétons.

Ils ont été introduits par [Dalal et Triggs, 2005] et s'inspirent fortement des travaux précédents de [Lowe, 2004] sur le descripteur SIFT (*Scale-invariant feature transform*). L'idée mise en œuvre ici est de discrétiser l'orientation du gradient dans l'image à l'aide d'histogrammes. Ainsi les contours, donc la forme des objets, sont codés ce qui permet de les reconnaître.

D'un point de vue pratique, le calcul d'un HOG commence par l'évaluation des gradients horizontaux G^x et verticaux G^y pour tous les pixels de l'image. Ce filtrage est réalisé à l'aide de deux convolutions effectuées séparément. La première utilise le masque $[-1,\ 0,\ 1]$ pour les gradients horizontaux et la seconde emploie le masque $[-1,\ 0,\ 1]^T$ pour les gradients verticaux. Ainsi à chaque point (i, j) de l'image est associé un couple $(G^x_{i,j},\ G^y_{i,j})$.

La norme n et l'orientation θ du gradient en un point sont alors définies par :

$$n_{i,j} = \sqrt{(G^x_{i,j})^2 + (G^y_{i,j})^2} \tag{2.8}$$

$$\theta_{i,j} = arctan(\frac{G^y_{i,j}}{G^x_{i,j}}) \tag{2.9}$$

L'image est ensuite décomposée en blocs, généralement de 8 pixels de côté, et un histogramme est construit pour chacun d'entre eux. Les classes des histogrammes représentent l'angle du gradient. Leur nombre est variable suivant les implémentations. [Dalal et Triggs, 2005] ont choisi d'en utiliser 9 dans l'intervalle $[0, \pi]$ (gradient non signé). Un pixel d'un bloc vote donc pour une classe en fonction de l'angle du gradient en ce point et son importance dans l'histogramme est la norme du gradient.

La dernière étape est la normalisation des histogrammes entre les blocs. Les histogrammes de blocs adjacents (par exemple des regroupements 4x4) sont concaténés pour former un vecteur qui est ensuite normalisé pour former une partie du descripteur. Le descripteur complet est obtenu en répétant l'opération sur tous les groupements de blocs possibles.

Il existe de nombreuses variantes par rapport aux HOG proposés initialement. Par exemple, [Ott et Everingham, 2009] intègrent une segmentation couleur de l'image pour donner plus de poids aux contours significatifs.

L'année suivante, [Dalal *et al.*, 2006] proposent le HOF (*Oriented Histograms of Flow*). Il s'agit d'une famille de descripteurs qui reprend les concepts éprouvés de HOG mais qui utilisent en entrée le signal de flot optique et non une image RGB. La procédure la plus simple pour exploiter ce signal est de reprendre la même approche que le HOG en calculant puis en répartissant les gradients de l'image de flot optique dans des histogrammes. [Dalal *et al.*, 2006] ont également proposé d'autres approches pour tirer partie du mouvement des différents morceaux d'un piéton.

2.3.1.5 Choix du descripteur dans nos travaux

Cette partie a permis d'évoquer quelques descripteurs parmi les plus importants rencontrés en détection de piétons. Pour nos travaux, nous avons choisi d'employer les descripteurs HOG pour coder l'apparence des piétons. En effet, ces derniers fournissent de bons résultats de classification tout en étant compatibles avec le temps réel et plus rapides à calculer que les matrices de covariances.

Nous allons maintenant présenter les différents algorithmes d'apprentissage employés en détection pour fabriquer les modèles de représentations des objets.

2.3.2 Apprentissage supervisé

2.3.2.1 Introduction à l'apprentissage statistique

L'apprentissage statistique est la brique qui va permettre de constituer une représentation de l'objet recherché (un piéton dans notre cas).

Les méthodes d'apprentissage travaillent à partir d'une base d'apprentissage, c'est-à-dire d'un ensemble de descripteurs labellisés (les exemples) ou non (les observations). L'utilisation de ces méthodes comporte généralement deux étapes distinctes : l'apprentissage proprement dit et la prédiction. La première phase consiste à étudier la structure des données d'apprentissage et à en dégager certaines informations comme les caractéristiques communes à des objets similaires ou appartenant à une même classe. La phase de

prédiction consiste à utiliser ces informations afin d'inférer des propriétés (type d'objets) des éléments d'une base d'exemples dite de test. Il est ainsi possible de prédire si une observation est un piéton ou non, en regardant si elle partage un certain nombre de caractéristiques avec les piétons présents dans la base d'apprentissage.

Il existe de nombreux types d'algorithmes d'apprentissage. Chacun répond à un type de problème particulier et à des *a priori* existants sur les données. Nous présentons d'abord brièvement les termes utilisés pour décrire les différents types d'apprentissage. Nous reviendrons sur les plus importants par la suite.

Classification : Le but de la classification est de déterminer le type (ou classe) d'un objet. Elle peut être **binaire** c'est-à-dire qu'il n'existe que deux classes possibles pour un objet (par exemple piéton et fond) ou **multiclasse** c'est-à-dire qu'il existe plusieurs catégories pour un objet (par exemple piéton, véhicule et fond). Dans cette thèse nous voulons discriminer les piétons du fond. Nous nous intéresserons donc exclusivement à la classification binaire.

Supervisé : Dans le cadre de l'apprentissage supervisé, les différentes classes possibles sont déjà connues. Pour fonctionner, l'algorithme a besoin d'un ensemble d'apprentissage contenant des exemples de toutes les classes avec leur label associé.

Non supervisé : En apprentissage non supervisé, les classes ne sont pas connues à l'avance. L'algorithme reçoit uniquement un ensemble d'observations. À lui ensuite de trouver, en fonction de la structure des données, les classes les plus pertinentes. Ces algorithmes peuvent être utilisés pour regrouper des observations proches en différents ensembles appelés *clusters*.

Semi-supervisé : L'apprentissage semi-supervisé est à mi-chemin entre les approches supervisées et non supervisées. Les classes sont connues lors de l'apprentissage, mais l'algorithme accepte en entrée des exemples qui ne sont pas forcément étiquetés.

Offline ou hors ligne : En apprentissage offline, tous les exemples sont connus dès le début. Il n'est plus possible par la suite d'en ajouter ou d'en enlever à moins de relancer toute la procédure. Cela permet d'avoir une vision globale du problème à traiter sous réserve d'être capable de stocker toutes les données. En contrepartie le classifieur obtenu est figé et ne peut donc pas s'adapter aux changements ou à une autre scène.

Online ou en ligne : Dans le cas online, les exemples sont obtenus les uns après les autres et servent à mettre à jour le classifieur de manière séquentielle. Une fois utilisé, un exemple est rejeté et ne sert plus pour la suite de l'apprentissage. Ces techniques peuvent traiter de grandes quantités de données, mais l'algorithme ne voit qu'une petite partie du problème à chaque fois. Il est donc plus difficile d'avoir un classifieur pertinent dans tous les cas de figures. Néanmoins ce dernier apprend continuellement et il est capable de s'adapter à de nouvelles conditions.

Batch : Un algorithme est en mode batch si à chaque itération (appelée aussi époque), tous les exemples d'apprentissage sont utilisés pour mettre à jour le classifieur.

Incrémental : Un algorithme est en mode incrémental si à chaque itération, seul un exemple est utilisé pour mettre à jour le classifieur. Cette manière de procéder n'implique pas forcément l'utilisation de méthodes online. Incrémental et online sont deux concepts différents.

Actif : L'apprentissage actif est une autre manière de traiter le cas où la base contient des données non labellisées. Dans l'ensemble d'apprentissage, toutes les observations n'apportent pas la même quantité d'information. Certaines observations sont plus importantes que d'autres, car elles sont plus proches de la frontière ou car elles sont dans une zone avec peu d'observations... L'apprentissage actif consiste à laisser à l'algorithme le choix de déterminer les exemples les plus pertinents parmi ceux qui ne sont pas labellisés. À charge ensuite à une entité externe, un humain par exemple, d'étiqueter les observations sélectionnées.

Transfert : Les algorithmes d'apprentissage font généralement l'hypothèse que les données d'apprentissage et de test sont dans le même espace et sont échantillonnées de manière indépendante et identiquement distribuées. Or cette hypothèse n'est pas toujours valide puisque les données peuvent provenir de scènes différentes. Le *transfer learning* tente de répondre à la problématique suivante : si nous disposons de deux ensembles d'apprentissage, l'un générique D_s et l'autre D_t représentant plus fidèlement les exemples de test, comment exploiter les connaissances contenues dans D_s pour améliorer la classification sur D_t ?

La suite de cette partie se concentre sur l'apprentissage supervisé. L'apprentissage semi-supervisé, actif et par transfert seront détaillés dans le chapitre 3 traitant de la contextualisation.

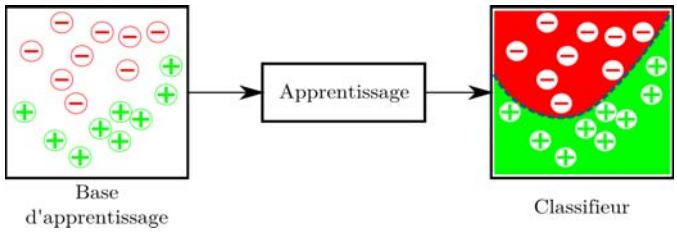
2.3.2.2 Principe de l'apprentissage supervisé

Les algorithmes d'apprentissage supervisé (*cf* figure 2.8(a)) sont adaptés au problème lorsque :

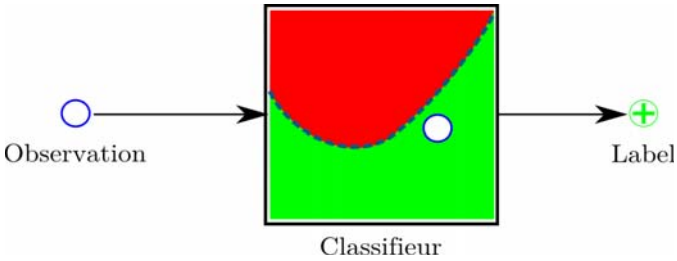
- les classes sont connues : piéton (+1) et fond (-1),
- les exemples d'apprentissage sont constitués de paires (descripteur / label).

Dans la phase d'apprentissage (*cf* figure 2.8(a)), un modèle de frontière est défini entre les descripteurs des deux classes d'objets. Dans la phase de prédiction (*cf* figure 2.8(b)), l'algorithme utilise la relation mise en évidence précédemment, pour inférer le label d'un descripteur appartenant à une classe inconnue. Nous verrons un peu plus loin (*cf* paragraphe 2.3.3) que pour détecter un piéton dans une image, il faut la parcourir et prédire à l'aide du classifieur, le label de chaque portion d'image étudiée.

De manière plus formelle, soit $\mathbb{X}$ l'espace des observations (ou l'espace des descripteurs) et $\mathbb{Y} = \{-1, 1\}$ celui des classes. Nous supposons que les couples $(x, y) \in \mathbb{X} \times \mathbb{Y}$ qui représentent les observations et leur classe associée, sont des variables aléatoires distribuées selon la loi P.



(a) Principe de l'apprentissage supervisé



(b) Principe de la prédiction

FIGURE 2.8 – Principe de fonctionnement de la classification supervisée.

Le rôle d'un algorithme d'apprentissage statistique est de construire une fonction (ou classifieur), h appartenant à l'ensemble des classifieurs $\mathbb{H}$, qui sert à prédire la classe y d'une observation x. La classe de x estimée par h est $\hat{y} = sgn(h(x)) \in \mathbb{Y}$.

$$\begin{aligned} h : \mathbb{X} &\longrightarrow \mathbb{R} \\ x &\longmapsto h(x), \text{ avec } sgn(h(x)) = \hat{y} \end{aligned} \tag{2.10}$$

La construction de h consiste en un problème d'optimisation. Un classifieur doit faire le moins d'erreurs possible sur ses prédictions. Pour cela nous introduisons une fonction de coût l (ou *loss function* en anglais). Cette dernière mesure la pénalité associée à l'événement $(h(x), y)$.

$$l : \mathbb{R} \times \mathbb{Y} \longrightarrow \mathbb{R}^+ \tag{2.11}$$

Il existe de nombreuses fonctions de coût possibles, comme par exemple la fonction caractéristique $\mathbb{1}_{\hat{y} \neq y}$. Généralement les fonctions convexes sont préférées car elles permettent de simplifier le problème d'optimisation.

Le critère à minimiser pour construire h est le risque R (ou erreur de généralisation) défini comme étant l'espérance de la fonction de coût :

$$\begin{aligned} R : \mathbb{H} &\longrightarrow \mathbb{R}^+ \\ h &\longmapsto E_{P(x,y)}[l(h(x), y)] = \int l(h(x), y) dP(x, y) \end{aligned} \tag{2.12}$$

Malheureusement, la probabilité P n'est pas connue. Le risque n'est donc pas mesurable directement. Pour contourner cette difficulté, h est optimisé à l'aide du risque empirique sur un échantillon D contenant n éléments de $(\mathbb{X}, \mathbb{Y})$:

$$R_{emp}(h, D) = \frac{1}{n} \sum_{(x,y)\in D} l(h(x), y) \tag{2.13}$$

D est appelé l'ensemble d'apprentissage (ou base d'apprentissage) et est constitué d'exemples positifs et négatifs. Un ensemble d'apprentissage adapté doit couvrir le maximum de cas différents afin que la relation définie par h soit le plus proche possible de la réalité.

L'hypothèse sous-jacente est que les exemples d'apprentissage et ceux de test sont distribués suivant les mêmes lois de probabilité. Elle n'est pas toujours respectée. Dans le cas de la détection, il est impossible de fabriquer un ensemble d'apprentissage qui couvre toutes les apparences possibles des piétons. Les bases construites sont alors spécifiques à telle ou telle situation et ne sont pas forcément représentatives des données de test.

Enfin entraîner un classifieur est un problème complexe car il est très difficile de trouver une séparation entre les classes qui soit à la fois valable sur l'ensemble d'apprentissage et généralisable à tout l'espace. En effet, il est possible, quelles que soient les données d'apprentissage, de trouver une fonction qui associe la bonne classe à tous les éléments de D, mais qui est incapable de généraliser au reste de l'espace $\mathbb{X}$. Dans ce cas, bien que le risque empirique soit minimal, le risque est élevé et les performances sont faibles. Ce phénomène s'appelle le sur-apprentissage et n'est pas souhaitable. Il peut survenir lorsque les frontières de décision du classifieur essaient de coller très fortement aux données. Par exemple, les observations bruitées ou proches de la séparation sont difficiles à distinguer et peuvent détériorer l'apprentissage si le classifieur y attache trop d'importance. Afin de ne pas découper l'espace abruptement autour de points aberrants et donc de limiter le pouvoir de généralisation du classifieur, les frontières doivent être modélisées de façon simple et régulière.

Puisqu'il est difficile de construire un bon classifieur sur tout l'espace, la principale contribution de cette thèse est la simplification du problème d'optimisation grâce à la contextualisation. Ainsi nous ne souhaitons pas optimiser un classifieur sur tout $\mathbb{X}$, mais seulement sur le sous-ensemble qui correspond à notre scène. D'autre part la contextualisation passe par la création, de manière automatique, d'un ensemble d'apprentissage compatible avec les données de test.

Parmi les algorithmes d'apprentissage supervisés, les deux principaux utilisés en vidéosurveillance sont les séparateurs à vaste marge (machines à vecteurs de support ou *Support Vector Machine*, SVM en anglais) et le boosting.

Support vector machine L'algorithme SVM a été initialement proposé par [Boser *et al.*, 1992; Vapnik, 1995]. Il appartient à la famille des algorithmes à noyau. Il a été utilisé à de nombreuses reprises dans la détection de piétons [Dalal et Triggs, 2005].

L'objectif des SVM est de trouver une séparation linéaire, c'est-à-dire un hyperplan, entre les exemples. La forme du classifieur h est donc l'équation d'un hyperplan :

$$< \omega, x > + \omega_0 = 0 \tag{2.14}$$

avec $<,>$ le produit scalaire euclidien.

Si les données sont linéairement séparables, il existe une infinité d'hyperplans séparateurs. Afin d'obtenir le pouvoir de généralisation optimal du classifieur, cet hyperplan doit maximiser la marge, c'est-à-dire la distance entre les exemples et la séparatrice linéaire. La marge d'un exemple (x, y) (positive en cas de bonne labellisation par le classifieur) est alors définie par sa distance à l'hyperplan :

$$M(x, y) = y\frac{(< \omega, x > + \omega_0)}{||\omega||} \tag{2.15}$$

Plus généralement, la marge M d'un exemple (x, y) pour un classifieur h est égale à $yh(x)$. Le critère pour entraîner un SVM est de maximiser la marge minimale (les vecteurs minimisant la marge sont appelés vecteurs supports) :

$$\arg\max_{\omega,\omega_0}\left(\frac{1}{||\omega||}\min_i[y_i(< \omega, x_i > + \omega_0)]\right) \tag{2.16}$$

Les données ne sont pas toujours linéairement séparables. Dans ce cas, il peut être judicieux de se placer dans un espace de dimension plus élevée (voire infinie) dans lequel les données peuvent être discriminées par un hyperplan. Cependant cela engendre des difficultés pour calculer le produit scalaire dans cet espace. Ce problème est résolu grâce au *kernel trick* où l'utilisation d'une fonction noyau permet de limiter les calculs dans un espace de dimension restreinte.

L'algorithme SVM standard souffre de deux inconvénients. Le premier est la lenteur relative de l'apprentissage. Pour pallier ce problème, des algorithmes basés sur la descente de gradient stochastique peuvent être utilisés [Bottou, 1998; Wijnhoven et de With, 2010].

L'autre difficulté vient du temps nécessaire à l'évaluation d'une hypothèse lors de la détection. L'algorithme de base n'est pas capable de faire de la sélection de composantes, c'est-à-dire d'expliciter les parties du descripteur qui sont pertinentes pour discriminer les deux classes. Or c'est très utile en détection temps réel car cela permet de ne pas calculer tout le descripteur et donc de gagner un temps précieux au moment de l'évaluation d'une hypothèse. Cette difficulté est une conséquence du fait que l'hyperplan appris n'est pas creux, c'est-à-dire que les nombreux coefficients de ω ne sont pas nuls. Des méthodes existent pour limiter ce phénomène. Par exemple, [Tan *et al.*, 2010] proposent d'utiliser un terme de régularisation creux.

Boosting offline Le boosting est un algorithme appartenant à la famille des apprentissages par ensemble (*ensemble learning*) qui comporte également le bagging. Le boosting est une méthode itérative introduite par Yoav Freund et Robert Schapire [Freund et Schapire, 1999; Schapire et Singer, 1999]. Contrairement aux SVM qui essaient de résoudre le problème dans sa globalité, le boosting tente de résoudre des problèmes plus simples, en combinant linéairement des classifieurs dits faibles pour construire une solution complète. En effet la prédiction de chaque classifieur faible est légèrement meilleure que le hasard, c'est-à-dire avec une erreur inférieure à 50%. L'intérêt est de prendre des classifieurs très simples et donc faciles à calculer. L'assemblage de nombreux classifieurs faibles forme alors un classifieur fort, dont les performances atteignent celles des SVM.

Historiquement les premiers algorithmes de boosting étaient supervisés et offline. Aujourd'hui des algorithmes online et semi-supervisés ont aussi été proposés. Parmi tous les algorithmes de boosting, le plus utilisé est probablement Adaboost (pour *adaptative boosting*) et ses variantes (RealAdaboost, LogitBoost [Friedman *et al.*, 1998; Schapire et Singer, 1999]). En 2001, [Viola et Jones, 2001b] ajoutent à l'algorithme classique une étape de sélection de composantes du vecteur descripteur ou *feature selection* (voir l'algorithme 1). Dans la version intiale d'Adaboost, un unique classifieur faible est entraîné à chaque itération. Viola et Jones décident d'entraîner tous les classifieurs faibles existants à chaque itération pour ne retenir que celui qui présente la plus faible erreur sur l'ensemble d'apprentissage. Une façon d'obtenir un ensemble de classifieurs faibles est de n'entraîner ces derniers que sur une partie du descripteur. Comme celui-ci est la plupart du temps calculé sur une grille, il est possible d'associer à chaque nœud de la grille un ou plusieurs classifieurs faibles. À l'origine, Viola et Jones ont appliqué cet algorithme à la détection de visages, puis plus tard, à la détection de piétons [Viola *et al.*, 2005].

Les classifieurs faibles peuvent être de natures différentes. L'idée du boosting est de s'appuyer sur des classifieurs très simples et les *decision stumps* [Ai et Langley, 1992; Viola et Jones, 2001b] sont souvent retenus. Un *decision stump* correspond à un étage d'un arbre de décision, c'est-à-dire à la comparaison d'une valeur à un seuil θ_s.

$$stump(x) = \begin{cases} a_1 & \text{si } x < \theta_s \\ a_2 & \text{sinon} \end{cases} \tag{2.20}$$

Ces classifieurs sont donc 1D. En combinant cette propriété avec le fait que le boosting sélectionne les meilleurs classifieurs faibles, il en ressort une méthode pour sélectionner les composantes du vecteur descripteur les plus pertinentes. L'entraînement d'un *decision stump* consiste à parcourir de manière exhaustive l'ensemble des seuils admissibles (les valeurs prises par les exemples) et de choisir le meilleur, c'est-à-dire celui qui minimise la borne supérieure de l'erreur d'entraînement (*cf* [Schapire et Singer, 1999]). Pour un exemple donné, plus la sortie du classifieur faible en valeur absolue est élévée, plus le classifieur est confiant dans sa prédiction. À noter que pour un Adaboost discret, a_1 et a_2 valent $+1$ ou -1 et que pour un Adaboost réel, les sorties a_1 et a_2 sont deux réels quelconques qui peuvent avoir des signes différents ou non.

La figure 2.9 montre l'architecture très simple d'un *decision stump*. La taille d'un exemple représente son poids. Bien que la meilleure frontière pour séparer les exemples

positifs et négatifs semble être plus à droite, en tenant compte des poids forts présents sur les exemples négatifs la démarcation est déplacée à gauche pour mieux les classifier.

Les *decision stumps* ne sont pas les seuls classifieurs faibles utilisés. [Zhu *et al.*, 2006] ont adapté le travail de [Dalal et Triggs, 2005] pour le boosting. Ils ont alors choisi

Algorithme 1: Adaboost offline discret avec sélection de composantes

Entrées :
Un ensemble d'apprentissage $D = \{(x_i, y_i)_{1 \leq i \leq m}, x_i \in \mathbb{X}, y_i \in \mathbb{Y}\}$,
n le nombre d'itérations (ou rounds) de boosting,
un ensemble $\mathbb{H}_f$ de classifieurs faibles

Sorties :
Un classifieur fort $h \in \mathbb{H}$, $\mathbb{H}$ étant l'ensemble des combinaisons linéaires des classifieurs faibles de $\mathbb{H}_f$

Algorithme :

Soit $\omega_{1,i} = 1/m$, le poids associé à l'exemple i à l'itération 1

pour $t = 1, \ldots n$ **faire**
Entraîner tous les classifieurs faibles, $h_j : \mathbb{X} \to \mathbb{R}$, de $\mathbb{H}_f$, en tenant compte de la distribution $\omega_{t,i}$
Calculer l'erreur $\epsilon_j = \sum_i \omega_{t,i} \mathbb{1}_{h_j(x_i) \neq y_i}$ de chaque classifieur faible h_j
Choisir le classifieur faible, nommé h_t, qui a l'erreur la plus faible ϵ_t
Affecter le poids $\alpha_t = 0{,}5 \ln(\frac{1-\epsilon_t}{\epsilon_t})$ à h_t
Mettre à jour les poids en diminuant ceux des exemples bien classifiés par h_t

$$\omega_{t+1,i} = \begin{cases} \omega_{t,i} e^{-\alpha_t} & \text{si } h_t(x_i) = y_i \\ \omega_{t,i} & \text{si } h_t(x_i) \neq y_i \end{cases} \tag{2.17}$$

Normaliser les poids $\omega_{t+1,i} = \frac{\omega_{t+1,i}}{\sum_j \omega_{t+1,j}}$

fin
Le classifieur fort est défini par :

$$h = \sum_t \alpha_t h_t \tag{2.18}$$

Soit θ le seuil de détection. Le label $\hat{y}$ d'un exemple x correspond au signe de $h(x) - \theta$ et vaut :

$$\hat{y} = \begin{cases} 1 & \text{si } \sum_t \alpha_t h_t(x) \geq \theta \\ -1 & \text{sinon} \end{cases} \tag{2.19}$$

$h(x)$ est une score de classification qui donne une indication sur la confiance du classifieur en sa prédiction.

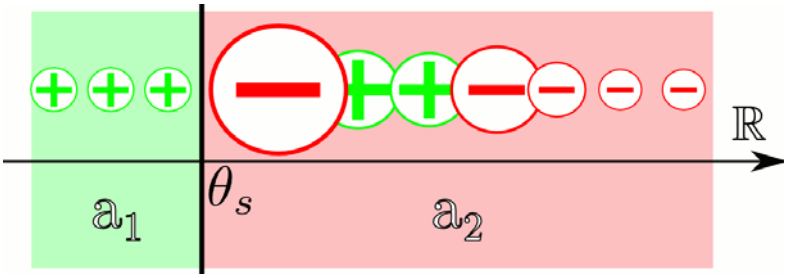


FIGURE 2.9 – Structure d'un *decision stump*.

de booster des SVM de petites dimensions. Néanmoins le couple boosting et *decision stumps* présente plusieurs avantages. Premièrement l'apprentissage est relativement rapide (seulement quelques minutes pour apprendre 10 000 exemples). Deuxièmement cette combinaison est souvent employée pour réaliser la sélection de composantes. Cette manière de faire permet de connaître les parties du descripteur qui sont importantes. Or, le choix du descripteur est une étape difficile et empirique. Une façon de contourner cette difficulté est d'employer un descripteur très grand en en concaténant plusieurs. En utilisant un algorithme de boosting, il est possible de ne sélectionner que les composantes pertinentes parmi cet ensemble. Cela permet de fabriquer un descripteur automatiquement.

Dans leurs travaux, [Mason *et al.*, 1999] ont démontré que la formation du classifieur fort par de nombreux algorithmes de boosting pouvait être considérée comme une descente de gradient dans l'espace des combinaisons linéaires des classifieurs faibles. Dans le cas d'Adaboost, la fonction de coût associée est $e^{-yh(x)}$, avec $yh(x)$ la marge de l'exemple.

2.3.2.3 Choix de l'algorithme d'apprentissage dans nos travaux

Dans cette thèse nous avons choisi de nous appuyer sur les algorithmes de boosting. En effet ces algorithmes fournissent de bons résultats et sont généralement rapides lors de l'apprentissage et lors de la détection. De plus, ils ne nécessitent pas de lourds réglages lors de l'entraînement. Néanmoins rien dans notre méthode n'est spécifique à ces algorithmes et même si nous ne l'avons pas testé, il devrait être possible de transposer notre approche avec d'autres algorithmes d'apprentissage.

Remarquons enfin que les algorithmes supervisés ont besoin pour travailler d'un ensemble d'apprentissage totalement labellisé. Pour notre application cela est problématique car nous souhaitons adapter le détecteur à chaque nouvelle caméra de manière automatique. Il est donc primordial de nous intéresser à d'autres méthodes d'apprentissage comme celles semi-supervisées qui prennent aussi en compte des données non labellisées. Toutes ces approches sont présentées dans le chapitre 3.5.

2.3.3 Mise en œuvre d'un détecteur

Nous venons d'étudier quelques méthodes d'apprentissage dont le but est de trouver une frontière entre des observations. Nous allons maintenant voir plus en détails comment ces algorithmes sont utilisés dans un détecteur.

2.3.3.1 Modèle de représentation des piétons

Pour résumer, les algorithmes d'apprentissage fabriquent des modèles de représentation des objets. Néanmoins, il est possible à partir de ces algorithmes de construire des modèles, monolithiques ou par parties, plus ou moins complexes.

Modèle monolithique Un classifieur monolithique est formé d'un modèle constitué d'un seul bloc. Son représentant le plus connu est certainement [Dalal et Triggs, 2005]. Dans ce cas, le classifieur reconnaît l'allure générale des piétons. Cette méthode est bien adaptée à la reconnaissance des piétons éloignés, ou ayant toujours la même allure. Typiquement en vidéosurveillance les piétons sont généralement debout.

Modèle par parties Un classifieur monolithique peut ne pas sembler optimal pour trouver des piétons qui sont par nature déformables. Les classifieurs par parties tentent de répondre à ce problème. Plusieurs systèmes ont été proposés comme ceux de [Wu et Nevatia, 2005] et [Felzenszwalb *et al.*, 2008]. Le but de ces approches est de fabriquer plusieurs classifieurs pour le même détecteur. Un classifieur est capable de reconnaitre un piéton en entier. Les autres plus spécialisés se focalisent sur des morceaux plus petits des personnes comme les bras, les jambes ou la tête. Lors de la détection, chaque classifieur est évalué indépendamment dans l'image. Si au voisinage d'un point, la plupart des classifieurs répondent fortement alors les chances pour qu'il s'agisse d'un piéton sont élevées.

2.3.3.2 Structure du classifieur

Comme nous l'avons vu précédemment avec [Dalal et Triggs, 2005], un classifieur peut être constitué d'un seul bloc. Dans ce cas l'évaluation du détecteur peut prendre du temps car il est nécessaire pour connaître la nature du descripteur de faire le calcul entièrement. Or dans un cas facile, par exemple un fond uni avec très peu de gradient, le calcul pourrait être abrégé. Différentes stratégies ont été mises en place pour pallier ce problème.

Cascade La cascade (voir la figure 2.10) est une structure particulière d'un classifieur dont le but est d'accélérer la détection. Le classifieur est découpé en différents étages. Chaque étage agit comme un classifieur complet. Un exemple est passé successivement dans tous les étages jusqu'à ce que l'un des classifieurs l'associe au fond. L'intérêt est que les premiers étages de la cascade sont très simples et qu'ils permettent d'éliminer rapidement les exemples les moins ambigus. Les derniers étages sont très spécialisés, plus complexes et donc plus lents. Ils permettent de traiter les exemples difficiles qui n'ont pas été arrêtés par les étages précédents. Cela permet de stopper l'évaluation d'un exemple une fois qu'il est certain qu'il s'agit du fond.

La cascade a été introduite par [Viola et Jones, 2001b] dans le cadre du boosting offline. Elle est basée sur le regroupement des composantes du descripteur. Un étage est constitué par un classifieur. Plus les étages avancent, plus le nombre de composantes pour construire un seul étage est grand. Il est donc plus coûteux d'évaluer le dernier étage par rapport au premier. Les premiers étages de la cascade sont constitués par les

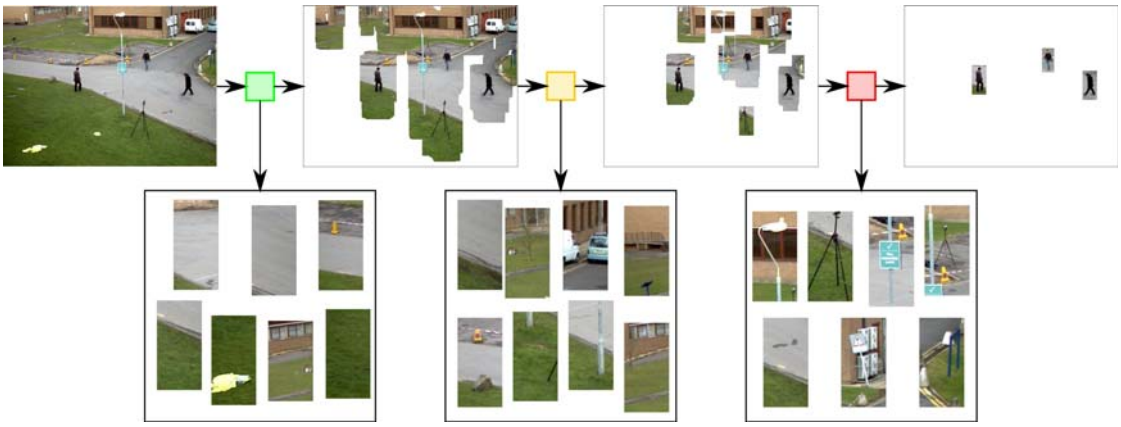


FIGURE 2.10 – Principe de la cascade. Le premier étage (en vert) est simple et élimine les exemples les plus faciles (zones uniformes avec peu de gradient). Les derniers étages plus complexes (orange et rouge) écartent des exemples plus difficiles ressemblant à des piétons comme les structures verticales (poteaux, arbres).

premières composantes sélectionnées par Adaboost et donc les plus pertinentes. Grâce aux premiers étages et donc avec très peu de classifieurs faibles, la majorité des observations du fond est arrêtée. Seuls les exemples représentant des piétons nécessitent l'évaluation complète de la cascade. De nombreuses variantes de cet algorithme ont été proposées pour traiter certains de ses problèmes comme le fait que les étages soient totalement indépendants [Bourdev et Brandt, 2005]. Toujours dans le cadre du boosting mais online cette fois-ci, [Visentini *et al.*, 2010] font l'hypothèse que les classifieurs faibles ayant une erreur faible permettent de mieux séparer le fond des objets. Ils ordonnent donc les classifieurs faibles du classifieur fort dans l'ordre croissant de leurs erreurs. De leur côté, [Wu et Nevatia, 2007b] proposent une solution incrémentale qui ré-entraîne partiellement une nouvelle cascade en fonction des données reçues en ligne.

Des cascades sont aussi utilisées avec des SVM. Pour diminuer les temps de calcul lors de la détection, il est possible de réduire le nombre de vecteurs supports [Romdhani *et al.*, 2004]. Une autre possibilité est de changer le type de noyau du classifieur entre chaque étage [Vedaldi *et al.*, 2009].

Arbre Une des limitations de l'algorithme de Viola et Jones est qu'il n'est capable de reconnaître qu'un seul type d'objets (en l'occurrence les visages vus de face) puisque la cascade ne possède qu'un seul nœud de sortie pour les objets à détecter. Cependant un même objet peut avoir différents aspects et faire un seul classifieur ne permet pas toujours de gérer tous les cas possibles. Les arbres [Tu, 2005; Wu et Nevatia, 2007a] sont une généralisation de la cascade dans laquelle un étage peut déboucher sur plusieurs nœuds de sorties. Un exemple ne passera donc pas par tous les classifieurs, mais uniquement par ceux qui lui sont spécifiques. Cela permet, lors de l'apprentissage d'entraîner chaque nœud, avec des exemples similaires. Le problème de la classification est donc simplifié localement et le détecteur dans son ensemble est capable de gérer des cas complexes.

À noter que [Babenko *et al.*, 2008] ont proposé MPLBoost (pour *Multiple Pose Learning*), une extension du boosting qui permet de le rendre plus robuste dans le cas où un même objet peut être observé sous des angles différents. Les exemples d'apprentissage

sont alors séparés en différents ensembles et un classifieur est entraîné sur chacun d'entre eux.

2.3.3.3 Stratégie d'exploration du détecteur

Le classifieur est une fonction qui prend en entrée un descripteur et qui fournit en sortie le label (piéton ou fond) associé. Pour connaître la localisation des personnes dans une image, il faut découper celle-ci en boîtes qui ne sont à ce stade que des hypothèses de recherche. Le classifieur les parcourt une à une et seules les hypothèses validées par le classifieur, c'est-à-dire celles contenant un piéton, sont conservées. De nombreuses stratégies ont été proposées dans la littérature pour définir les hypothèses. Elles dépendent de la structure du classifieur et de la scène.

Fenêtre glissante La méthode de la fenêtre glissante [Viola et Jones, 2001b; Dalal et Triggs, 2005] est probablement l'approche la plus classique. Elle consiste à balayer l'image, de manière régulière, avec le détecteur.

Un **parcours 2D** peut être effectué (figure 2.11(a)) sans connaissance *a priori* de la scène et donc sans connaissance des dimensions de la boîte en un endroit donné de l'image. Le parcours se fait, à des échelles différentes, de manière dense dans le plan de l'image. Il est parcouru plusieurs fois avec des boîtes de tailles différentes de manière indépendante. En 2D, une façon de générer des boîtes de tailles (largeur x hauteur) différentes est de partir d'une taille de base $t_0 = l_0$ x h_0 (classiquement 64 x 128). La taille s, est obtenue en multipliant s fois la largeur et la hauteur de la boîte par un facteur d'échelle α : $t_s = l_0\alpha^s$ x $h_0\alpha^s$.

Un **parcours 3D** (figure 2.11(b)) peut être défini à condition de disposer d'une caméra calibrée, c'est-à-dire de connaître ses paramètres internes (focale, centre optique) et externes (position et rotation de la caméra par rapport au repère monde) (voir [Hartley et Zisserman, 2004] pour plus de détails).

Le sol, supposé plan, est échantillonné afin de définir la position de toutes les hypothèses dans le repère monde. Alors, sachant que les piétons se déplacent sur ce plan et que la taille standard d'un être humain est d'environ 1,80m, il est possible d'en déduire la taille et l'orientation de la silhouette générale d'une personne en chaque point de l'image. Le piéton est modélisé dans l'espace 3D et cette forme est projetée dans le plan image. Un modèle ellipsoïdal est une bonne approximation et sa projection dans le plan image qui forme une ellipse, est facilement calculable [Eberly, 1999]. L'hypothèse testée en ce point de l'image par le détecteur est alors constituée par la boîte circonscrite à cette ellipse avec éventuellement une marge autour de celle-ci. Pour construire l'ensemble des hypothèses, la procédure est répétée pour tous les points du plan du sol.

Dans le cas d'une caméra fixe, les hypothèses 3D étant relativement peu nombreuses mais coûteuses à obtenir, il est souhaitable de les précalculer à l'initialisation.

Ainsi contrairement au cas 2D où il est nécessaire de faire beaucoup d'hypothèses lors du parcours du détecteur dans l'image puisque de nombreux paramètres sont inconnus, il est possible en 3D de ne tester que les boîtes intéressantes et ainsi d'augmenter fortement la rapidité de la détection.

(a) Stratégie de parcours 2D. Les hypothèses sont générées par des boîtes de tailles différentes qui sont déplacées de manière dense dans l'image.

(b) Stratégie de parcours 3D. Le plan du sol est échantilloné de manière dense (point). En chaque point, une hypothèse est définie en fonction de la taille et de l'orientation d'un piéton potentiel à cet endroit. Pour des raisons de clarté, seules quelques hypothèses sont représentées.

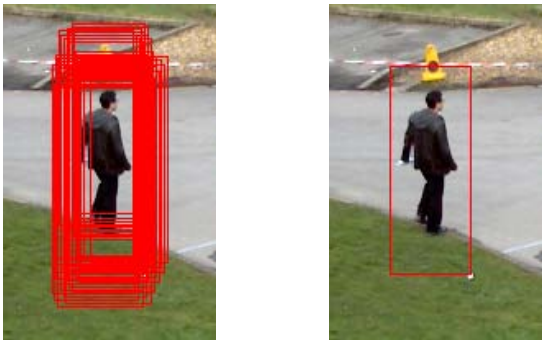
FIGURE 2.11 – Différentes stratégies de parcours du détecteur dans l'image à l'aide de la méthode des fenêtres glissantes.

Tirage particulaire Une autre approche a été proposée par [Gualdi *et al.*, 2010]. Dans le cas où l'image traitée est de grande taille, il n'est plus possible d'utiliser une méthode par fenêtre glissante à moins d'augmenter le pas de manière importante. L'idée est alors d'effectuer une détection par raffinements successifs. Dans une première étape, l'image est échantillonnée uniformément (dans le cadre d'une vidéo, il est également possible d'utiliser les positions antérieures connues des piétons pour affiner la recherche) avec un pas assez grossier. À l'itération suivante les nouvelles positions (ou particules) sont tirées en fonction des scores obtenus par les hypothèses testées précédemment. Plus les particules de l'étape $n-1$ ont un score élevé plus le tirage des particules va se concentrer dans cette zone.

Grille de classifieurs Les grilles de classifieurs (*classifier grids*), [Grabner *et al.*, 2007; Roth *et al.*, 2009; Stalder *et al.*, 2009b; Sternig *et al.*, 2010], sont employées dans le cas d'une caméra fixe et stationnaire de type vidéosurveillance. Cette méthode a pour but de contraindre spatialement le problème de la détection d'objets. Les grilles de classifieurs sont constituées d'un classifieur fixe en chaque point de l'image. Un classifieur appartenant à une grille traite donc toujours les mêmes événements et est spécialisé pour reconnaître les objets sous une certaine apparence dans une fenêtre de l'image. Cela évite de déplacer, dans toute l'image, un classifieur plus complexe qui doit reconnaître les nombreuses apparences d'un même objet et permet de simplifier l'étape de classification.

2.3.3.4 Post-traitements : regroupement des détections

Le parcours du détecteur dans l'image fournit un ensemble de boîtes brutes (figure 2.12(a)), constitué de toutes les boîtes qui ont un score de classification supérieur au seuil de détection. Un classifieur a toujours une réponse floue. Ainsi pour un même piéton, il est normal que plusieurs boîtes, espacées de quelques pixels seulement, répondent fortement. Pour présenter les résultats à un opérateur ou pour utiliser les détections dans un autre algorithme (par exemple du tracking), il est nécessaire de regrouper les détections afin d'obtenir une boîte englobante par piéton (figure 2.12(b)).



(a) (b)

FIGURE 2.12 – Les post-traitements assurent la fusion des détections brutes 2.12(a) en une seule 2.12(b)

Pour cette tâche un algorithme non supervisé et non paramétrique est utilisé, c'est-à-dire un algorithme qui ne prend pas le nombre de clusters en paramètre. En effet le nombre de piétons étant inconnu dans l'image, il est important que le regroupement soit capable de déterminer le nombre de clusters optimal. Les deux algorithmes les plus populaires sont le mean shift et la suppression des non maximums.

Mean shift Le mean shift [Fukunaga et Hostetler, 1975; Cheng, 1995; Comaniciu *et al.*, 2002] est un algorithme très employé en vision par ordinateur. Il a été utilisé avec succès en segmentation d'image, tracking et a été étendu par [Dalal, 2006] pour le regroupement des hypothèses validées par un détecteur d'objet.

L'idée est d'estimer la fonction de densité du paramètre étudié pour en trouver les points maximaux. Ici le paramètre est la répartition des boîtes détectées et l'algorithme du mean shift va estimer la fonction de densité des boîtes. Les zones importantes sont les plus denses car ce sont elles qui ont la plus grande probabilité de contenir une détection.

Les calculs mentionnés ci-dessous s'applique dans le cas d'un mean shift dans le plan de l'image. Soit x_i, y_i l'abscisse et l'ordonnée du centre et s_i l'échelle de la boîte i, il y a n boîtes à fusionner et chacune est représentée par le vecteur $B_i = \begin{pmatrix} x_i \\ y_i \\ s_i \end{pmatrix}$ et associée avec son score de classification θ_i supposé positif. Le mean shift utilise dans un premier temps l'estimation par noyau (fenêtres de Parzen [Parzen, 1962]) pour déterminer la densité des boîtes. Généralement, pour regrouper les boîtes, un noyau gaussien est utilisé.

$$K_H(B) = \frac{1}{(2\pi)^{\frac{3}{2}}} |H|^{\frac{-1}{2}} e^{-\frac{1}{2} B^T H^{-1} B} \tag{2.21}$$

$$\tag{2.22}$$

avec H la matrice diagonale $diag[h_x, h_y, h_s]$ qui est le paramètre du noyau dans les trois dimensions. Ces paramètres, positifs, indiquent l'incertitude sur la position des boîtes et influent sur le lissage de l'estimation. Plus ils sont élevés, moins la position de la boîte est fiable et plus le noyau et donc l'estimation de densité sont uniformes et apportent donc peu d'information. Cette incertitude varie également en fonction de l'échelle de la boîte détectée. En effet, plus une boîte est petite, plus elle devrait être localisée de manière précise par le détecteur. Les paramètres H dépendent donc de la boîte considérée.

L'estimation de la fonction de la densité D des boîtes en un point $B = (x, y, s)^T$ est obtenue en associant, pour chaque boîte, une gaussienne, pondérée par la confiance du classifieur en cette boîte. Les gaussiennes sont ensuite sommées.

$$\begin{aligned} D : \mathbb{R}^3 &\longrightarrow \mathbb{R}^+ \\ B &\longmapsto \frac{1}{n} \sum_{i=1}^{n} \theta_i K_{H_i}(B - B_i) \end{aligned} \tag{2.23}$$

Les boîtes regroupées correspondent aux maximums de la fonction de densité. Ils sont obtenus grâce à une montée de gradient opérée au départ de chaque boîte. Idéalement, toutes les hypothèses validées par le détecteur autour d'un même piéton converge vers la même boîte centrée sur ce dernier.

L'algorithme du gradient est une méthode itérative qui permet d'estimer les maximums locaux d'une fonction f. Un point X^0 est successivement translaté dans la direction du gradient $X^{m+1} = X^m + \alpha \nabla f(X^m)$ jusqu'à convergence vers un point où la dérivée de f s'annule. Dans le cadre du regroupement des hypothèses, chaque boîte B est déplacée par le vecteur mean shift $m(B)$, colinéaire au gradient de la densité $D(B)$ et défini par :

$$m(B) = \frac{H(B)}{D(B)} \nabla D(B) = H(B)[\sum_{i=1}^{n} \varpi_i(B) H_i^{-1} B_i] - B \tag{2.24}$$

avec

$$H(B) = \frac{1}{\sum_{i=1}^{n} \varpi_i(B) H_i^{-1}} \tag{2.25}$$

et

$$\varpi_i(B) = \frac{|H_i|^{-\frac{1}{2}}\ \theta_i\ exp(-\frac{(B-B_i)^T H^{-1} (B-B_i)}{2})}{\sum_{i=1}^{n} |H_i|^{-\frac{1}{2}}\ \theta_i\ exp(-\frac{(B-B_i)^T H^{-1} (B-B_i)}{2})} \tag{2.26}$$

La procédure du mean shift est résumée par l'algorithme 2.

Algorithme 2: L'algorithme du mean shift pour regrouper les boîtes.

Entrées :
Un ensemble **B** contenant n boîtes, B_i, issues de la détection avec leur score de classification associé
Le nombre d'itérations maximales $maxIt$

Sorties :
Un ensemble de détections regroupées et stockées dans **B**

Initialisation :
Copier **B** pour créer l'ensemble $\mathbf{B}^0 = \{B_i^0\}_{1 \leq i \leq n}$
$\mathbf{B}^0$ n'est pas modifié par l'algorithme et sert à l'estimation de la densité de boîtes.

Algorithme :

répéter
  pour chaque *boite* $B_i \in \mathbf{B}$ **faire**
    Calculer le vecteur mean shift $m(B_i)$ à l'aide des boîtes $\mathbf{B}^0$.
    Mettre à jour **B** en déplaçant B_i de $m(B_i)$,
    c'est-à-dire remplacer B_i par $H(B_i)[\sum_{j=1}^{n} \varpi_j(B_i) H_j^{-1} B_j^0]$.
  fin
  Supprimer les boîtes redondantes dans **B**
jusqu'à *stabilisation de* **B** *ou dépassement du nombre d'itérations* $maxIt$;

Suppression des non maximums Un des inconvénients du mean shift est qu'il demande une grande quantité de calculs. De plus une boîte obtenue après un regroupement ne coïncide pas forcément avec une hypothèse testée par le classifieur.

L'algorithme 3 corrige ces deux défauts. Il utilise une métrique entre deux boîtes. Un critère de similarité souvent employé est celui choisi par le challenge PASCAL([Everingham *et al.*, 2009]) :

$$sim(B_1, B_2) = \frac{Aire(B_1 \cap B_2)}{Aire(B_1 \cup B_2)} \tag{2.27}$$

Deux boîtes sont considérées comme étant similaires si $sim(B_1, B_2) > 0,5$.

2.3.3.5 Choix pour la mise en œuvre du détecteur dans nos travaux

Dans ce paragraphe sont présentées les principales caractéristiques des détecteurs mis en œuvre au cours de cette thèse. Le but de nos travaux est de mesurer l'apport de la contextualisation sur le détecteur. En ce sens, nous avons préféré travailler sur une structure simple afin de mieux comprendre les interactions entre le détecteur et la méthode de contextualisation étudiée. D'autre part, l'objectif de cette thèse est de déboucher sur un système contextualisé totalement fonctionnel à moyen terme. Inévitablement, un tel système doit pouvoir fonctionner en temps réel ce qui influe nécessairement sur la

Algorithme 3: Suppression des non maximums

Entrées :
Un ensemble **B** de boîtes issues de la détection avec leur score associé θ_i
Une similarité sim entre deux boîtes et un seuil s_c

Sorties :
Un ensemble R de détections regroupées

Algorithme :

Trier la liste **B** des détections par ordre de score θ_i décroissant.
Initialiser la liste R des regroupements à $\emptyset$

tant que $B \neq \emptyset$ **faire**
  Sélectionner la première détection B de **B** pour former un nouveau regroupement
  Supprimer de **B** toutes les détection B_i telle que $sim(B, B_i) > s_c$
  Ajouter B à R
fin

structure même du détecteur. Il a donc fallu parfois réaliser des compromis entre simplicité et rapidité du détecteur.

Concrètement nos détecteurs utilisent un modèle monolithique, sans cascade et leur parcours dans l'image est basé sur le formalisme des fenêtres glissantes.

Les expérimentations peuvent reposer sur un détecteur aussi bien 2D que 3D. La génération des hypothèses ainsi que les post-traitements sont différents dans les deux cas. Pour un détecteur 2D, de nombreuses boîtes, souvent à des échelles différentes et pas toujours centrées sur la personne, doivent être regroupées. Le mean shift, algorithme robuste, est bien adapté à cette tâche. Pour un détecteur 3D, le nombre d'hypothèses testées est plus faible, il y a donc moins de boîtes à regrouper. L'algorithme de suppression des non maximums est plus adapté dans ce cas car il est plus rapide.

2.4 Conclusion

Les méthodes de détection de piétons basées sur la classification sont aujourd'hui très populaires. Malgré leurs besoins plus importants en terme de calculs que les méthodes basées sur la temporalité, elles fournissent de bons résultats et sont mieux adaptées dans les situations où plusieurs types d'objets sont en mouvement. Cependant ces méthodes sont sensibles car elles dépendent de la base d'apprentissage. En particulier, elles ont du mal à s'adapter à des situations inconnues. Une différence de point de vue entre la base d'apprentissage et la scène peut provoquer une chute non négligeable des performances. Cela peut être problématique lorsqu'il faut déployer un système de surveillance car il est impossible pour un opérateur humain d'annoter une base d'apprentissage pour chaque élément du réseau.

L'idée qui sera détaillée dans le chapitre 3, consiste à introduire directement au sein du détecteur des informations provenant de la scène sur laquelle il va être employé.

Partie I : État de l'art

3

Contextualisation d'un détecteur

Sommaire

3.1 Introduction 42
3.2 Limites des détecteurs génériques 42
3.3 Évaluation d'un détecteur de piétons 44
3.3.1 Les courbes précision-rappel 45
3.3.1.1 Définitions 45
3.3.1.2 Construction 45
3.3.2 Données d'évaluations 47
3.3.3 Réflexions sur la création de la vérité terrain 49
3.4 Travaux préliminaires justifiant la contextualisation 51
3.5 Apprentissage pour la contextualisation 53
3.5.1 Apprentissage semi-supervisé 53
3.5.2 Approches avec oracles 56
3.5.3 Apprentissage actif 60
3.5.4 Apprentissage par transfert 62
3.5.4.1 Transfert sur les exemples 62
3.5.4.2 Transfert sur les composantes 63
3.6 Conclusion 63

3.1 Introduction

Le chapitre précédent a présenté les méthodes génériques de détection de piétons et notamment celles basées sur la reconnaissance de formes. Néanmoins, ces approches possèdent un certain nombre de limites qui sont illustrées dans la partie 3.2 de ce chapitre. En particulier, dès que l'apparence des exemples de la base d'apprentissage et de ceux de la scène observée diffèrent, les performances chutent.

Ce problème peut être évité grâce à la contextualisation du détecteur qui consiste à tenir compte dans le détecteur des spécificités de la scène. La contextualisation est un sujet vaste car de nombreux composants du détecteur peuvent bénéficier de connaissances concernant les futures conditions d'exploitation. Généralement, l'une des premières sources d'information exploitée est l'apparence des piétons qui permet au travers d'une base d'apprentissage dite contextualisée d'agir sur le classifieur.

La partie 3.3 expose le protocole expérimental choisi pour mesurer l'apport de la contextualisation du détecteur, ainsi que les jeux de données qui servent à l'évaluation de tous les algorithmes dans ce mémoire.

Dans la suite du chapitre, différentes façons de contextualiser un détecteur sont étudiées. Leur but est de réduire les problèmes rencontrés par les approches génériques. Nos travaux préliminaires, présentés dans le paragraphe 3.4, démontrent que la contextualisation d'un détecteur est bénéfique pour les performances. La partie 3.5 aborde différents algorithmes d'apprentissage qui peuvent se révéler utiles pour rendre le système automatique et résoudre le problème de la contextualisation. Ils viennent en complément de l'apprentissage supervisé évoqué précédemment.

3.2 Limites des détecteurs génériques

Les détecteurs de piétons actuels présentent de bonnes performances globales mais sont encore perfectibles. Ces limites résultent de causes variées. Les erreurs sont de deux types. Les faux positifs, aussi appelés fausses détections, sont des boîtes qui ne contiennent pas de piéton et les faux négatifs correspondent aux piétons qui ne sont pas repérés. Les images suivantes illustrent au travers de quelques situations, les cas les plus fréquents d'échecs des détecteurs actuels. En rouge figurent les bonnes détections. Les rectangles verts représentent les faux positifs tandis que les ellipses bleues indiquent des faux négatifs.

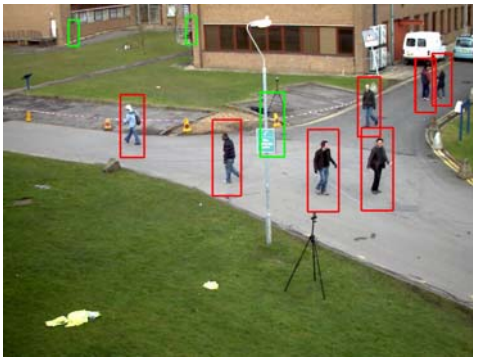


FIGURE 3.1 – Un détecteur de piétons se base généralement sur les contours des éléments de la scène pour déterminer s'il s'agit ou non d'un piéton. Dans le cadre de la vidéosurveillance et compte tenu du positionnement des caméras, les personnes sont observées de manière relativement droite et forment ainsi des structures verticales. Il arrive alors que le détecteur confonde certaines structures verticales (en vert sur l'image) comme les rebords de fenêtres, les poteaux et les arbres avec des piétons.

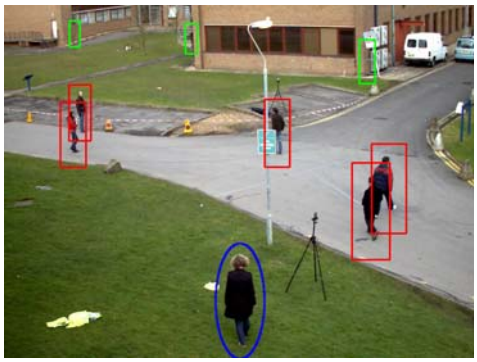


FIGURE 3.2 – Un détecteur ne reconnaît pas uniquement un piéton, il considère aussi son environnement proche. En conséquence les boîtes de détections (rouge) fournies sont plus grandes que le piéton lui-même. Lorsque l'environnement n'est pas connu, le détecteur ne fonctionne pas correctement. La personne contenue dans l'ellipse bleue est trop proche du bord de l'image et n'est pas détectée.

FIGURE 3.3 – L'ellipse de droite illustre un problème lié aux post-traitements. Seulement deux piétons sur trois sont repérés car ils sont trop proches les uns des autres et leurs détections ont été fusionnées. À gauche, les deux piétons entourés sont difficiles à discerner car ils se confondent aisément avec le fond. Un détecteur de piétons fonctionne généralement image par image et n'a donc pas conscience de la dynamique de la scène et du mouvement des objets, ce qui rend ce type de piéton difficile à détecter.

FIGURE 3.4 – Il arrive que des détections fantômes apparaissent près d'autres détections (boîte verte). Cela peut se produire quand plusieurs zones proches ont un score élevé comme plusieurs piétons ensemble ou un piéton approchant d'une structure parasite (poteau). Les post-traitements peuvent être en difficulté pour fusionner correctement toutes les boîtes.

FIGURE 3.5 – Le piéton entouré n'est pas détecté à cause des déformations dues à la perspective. En effet, contrairement aux autres piétons, celui-ci n'est plus vertical mais penché. Cette image est hétérogène. La détection sur la partie gauche fonctionne relativement bien, mais pas sur la partie droite où il y a de nombreuses fausses détections (en vert). Le fond de l'image est complexe dans cette zone contenant des contours parasites. Cela signifie que le classifieur n'y est pas adapté. Le seuil de détection est manifestement trop bas.

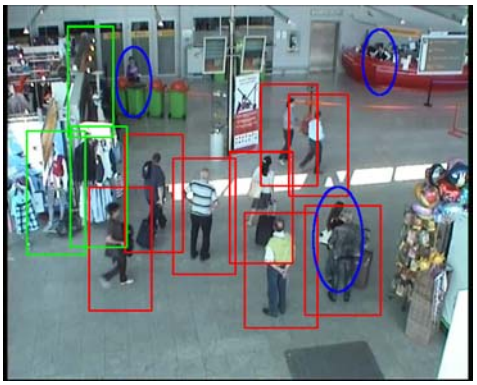


FIGURE 3.6 – Cette image illustre deux difficultés. Au fond et au premier plan, trois piétons (ellipses) sont partiellement occultés et ne sont donc pas détectés. Au premier plan (boîtes vertes), sont exposés des vêtements qui ressemblent à des humains. Le détecteur ne comprend pas la scène et ne peut pas faire la différence entre un vêtement sur un présentoir et un autre sur un humain.

La contextualisation d'un détecteur peut aider à résoudre la plupart de ces problèmes de manière directe ou indirecte. Ce processus devrait conduire à une meilleure prise en compte de l'apparence des piétons de la scène. Ainsi le système devrait être *a priori* capable de les détecter même si le point de vue présente des difficultés liées à la perspective. Ensuite, les faux positifs sont généralement bien moins nombreux après le processus de contextualisation. En effet, ils sont normalement intégrés dans la base des exemples négatifs et idéalement le classifieur contextualisé les a appris comme tels. Le système a donc appris à les reconnaître correctement et se trompe moins souvent. Enfin, en ce qui concerne les problèmes liés aux post-traitements et notamment les regroupements des détections, bien que la contextualisation n'apporte pas de solution immédiate à ce défaut, elle permet de l'atténuer. En effet, les réponses d'un détecteur contextualisé se font plus précises et plus ponctuelles. Le travail de l'algorithme de regroupement s'en trouve alors facilité.

3.3 Évaluation d'un détecteur de piétons

Constater l'existence de certaines limites, comme dans le paragraphe précédent, ne suffit pas pour caractériser les performances d'un détecteur. La mise au point d'une technologie nécessite obligatoirement une évaluation rigoureuse. Pour se rendre compte des performances d'un détecteur par rapport à un autre, il faut réaliser des comparaisons précises entre les algorithmes initiaux et leurs améliorations potentielles.

Cette partie explique les choix qui ont été faits durant cette thèse pour analyser et mesurer les performances des détecteurs. Nous présentons d'abord l'outil de comparaison proprement dit : les courbes précision-rappel, puis les données employées tout au long de ce travail.

3.3.1 Les courbes précision-rappel

3.3.1.1 Définitions

Il existe de nombreux indicateurs couramment employés pour mesurer les performances d'un classifieur binaire. Une possibilité consiste à tracer une courbe ROC (*Receiver Operator Characteristic*) qui représente le taux de vrais positifs en fonction du taux de faux positifs. La détection de piétons est un problème asymétrique, dans lequel le nombre de négatifs est inconnu et très supérieur au nombre de positifs, ce qui rend l'évaluation du taux de faux positifs difficile. Dans ce cas, les courbes précision-rappel semblent mieux adaptées. [Davis et Goadrich, 2006] ont fait une analyse comparative des deux représentations.

Par définition, le rappel R est la fraction de piétons correctement détectés parmi tous les piétons. La précision Pr est la fraction de détections correctes parmi toutes les détections. Soient VP le nombre de vrais positifs, FP le nombre de faux positifs et P le nombre de piétons, la précision et le rappel peuvent être exprimés comme suit :

$$R = \frac{\text{nombre de détections correctes}}{\text{nombre de piétons}} = \frac{VP}{P} \tag{3.1}$$

$$Pr = \frac{\text{nombre de détections correctes}}{\text{nombre de détections}} = \frac{VP}{VP + FP} \tag{3.2}$$

3.3.1.2 Construction

Sauf mention contraire, toutes les courbes de cette thèse ont été réalisées à l'aide du protocole expérimental suivant, inspiré par les travaux de [Agarwal *et al.*, 2004].

Nous traçons les courbes paramétriques représentant R en fonction de $(1 - Pr)$ en faisant varier le seuil de détection. Le rappel et la précision sont évalués avant l'application d'un algorithme de regroupement de boîtes, ce qui permet de n'évaluer que la performance du classifieur et non celle des post-traitements. Une courbe atteignant le point situé en (0, 1) est idéale et signifie que le détecteur est parfait. Un détecteur réglé avec un seuil bas peut repérer l'ensemble des piétons mais sa réponse risque de contenir de nombreux faux positifs. Son rappel sera donc élevé mais sa précision faible. Un seuil haut va générer l'effet inverse, un faible rappel et une précision élevée.

Pour régler convenablement le détecteur, il est important de trouver un compromis entre le rappel et la précision. La F-Mesure en est un qui considère que rappel et précision ont autant d'importance. Elle est définie par la relation suivante :

$$FM = 2.\frac{Pr.R}{Pr + R} \tag{3.3}$$

Il s'agit de la moyenne harmonique du rappel et de la précision. Plus cette valeur est grande, meilleur est le classifieur. Le point optimal d'un classifieur correspond donc au seuil qui maximise la F-Mesure.

Pour pouvoir évaluer les résultats d'un algorithme, il faut disposer d'une vérité terrain qui sert de référence et qui contient la liste des détections idéales de la séquence. Elle est généralement construite à la main.

Pour comparer une boîte de la vérité terrain (VT) et une boîte (B) obtenue avant regroupement, nous avons retenu le critère de similarité suivant :

$$sim(VT, B) = \frac{(VT_{cx} - B_{cx})^2}{(0.5 \times l(VT))^2} + \frac{(VT_{cy} - B_{cy})^2}{(0.5 \times h(VT))^2} \tag{3.4}$$

avec :

- cx et cy respectivement l'abscisse et l'ordonnée du centre de la boîte,
- $l(VT)$ et $h(VT)$ la largeur et la hauteur de la boîte de la vérité terrain.

Deux boîtes sont similaires si $sim(VT, B) \leq 1$. L'avantage de ce critère est qu'il a une signification physique. En effet, une boîte est similaire à une autre si son centre est contenu dans un ellipsoïde situé autour du centre de cette seconde. Pour déterminer VP, FP et ainsi en déduire le rappel et la précision, il a été décidé d'employer la métrique suivante qui est représentée sur la figure 3.7 :

- si le centre d'une détection est contenu dans l'ellipsoïde alors elle est considérée comme valable,
- si deux détections sont associées à la même boîte de la vérité terrain nous ne comptabilisons qu'une bonne détection,
- les autres boîtes correspondent à des fausses détections.

Ainsi, un positif n'est comptabilisé qu'une seule fois, même si après l'étape de post-traitements deux boîtes en résultent.

(a) (b)

FIGURE 3.7 – Métrique pour comparer les détections à la vérité terrain. (a) Ces deux boîtes n'ont pas leur centre dans l'ellipse. Il s'agit de deux faux positifs. (b) Toutes ces boîtes ont leur centre dans l'ellipse. Il s'agit d'une bonne détection.

Parfois, il peut être intéressant de connaître les performances moyennes d'un algorithme. Dans ce cas, l'expérience est réalisée plusieurs fois sur la même vidéo et avec les mêmes paramètres (au moins 5 fois) puis moyennée. Le rappel moyen est calculé pour une précision donnée. Les seuils de détections sont donc différents d'un classifieur à l'autre. La variance est obtenue selon le même principe et elle est affichée de part et d'autre de la courbe moyenne à l'aide de barres d'erreur à plus ou moins 2σ.

3.3.2 Données d'évaluations

Une évaluation commence par un choix judicieux des données employées. L'exercice de labellisation de celles-ci ne doit être ni trop facile ni trop difficile pour les capacités du détecteur afin de discriminer convenablement les différentes approches expérimentées.

La contextualisation d'un détecteur de piétons impose un certain nombre de caractéristiques pour les vidéos de test. La caméra doit être fixe et la scène filmée pendant suffisamment longtemps pour que la collecte des données nécessaires à la contextualisation soit possible (lors de nos tests, nous avons éliminé les séquences dont la durée était inférieure à 5 minutes). Afin de montrer l'apport de la contextualisation, l'angle de vue ne doit pas être exactement le même dans la base d'apprentissage et dans la base de test.

Voici les différents jeux de données que nous avons employés durant cette thèse.

INRIA Person Dataset

La base de piétons de l'INRIA contient un ensemble d'images divisé en deux catégories : celles contenant des piétons pris à hauteur d'homme (exemples positifs) et celles ne contenant que du fond (exemples négatifs). Nous avons utilisé cette base pour entraîner tous les classifieurs génériques. La figure 3.8 présente un échantillon de la base.

Les imagettes labellisées positivement ont une largeur de 64 pixels et une hauteur de 128 pixels avec des marges autour des piétons. Pour les exemples négatifs, les images étant plus grandes, une étape de préparation est nécessaire pour les découper à la même taille que les positives. Cela influe sur le type de vidéos que le système de contextualisation est capable de traiter. En particulier les tailles des piétons doivent être du même ordre de grandeur. Notre système ne sera pas capable de détecter des piétons faisant seulement 5 à 10 pixels de haut. Une autre base serait probablement nécessaire dans ce cas. L'INRIA Person Dataset est disponible à l'adresse suivante : http://pascal.inrialpes.fr/data/human/.

(a) (b) (c) (d)

FIGURE 3.8 – Exemples d'images provenant de la base INRIA

PETS

Les séquences PETS (*Performance Evaluation of Tracking and Surveillance*) sont publiques et ont été réalisées dans le cadre de workshops pour permettre aux chercheurs de comparer leurs algorithmes suivant un protocole commun. Ces séquences sont découpées en plusieurs vidéos d'une durée approximative de 3 minutes chacune.

PETS 2006

La séquence PETS 2006, http://www.cvg.rdg.ac.uk/PETS2006/, a été filmée dans une gare. Plusieurs vues sont disponibles, nous avons utilisé la vue 4. Cette scène est prise en légère plongée. Les effets de la perspective sont limités aux variations d'échelle. En conséquence, les piétons ont une apparence relativement proche de celle proposée par la base INRIA. L'éclairage de cette scène engendre de nombreux reflets sur le sol mais aussi sur les vitres du quai, ce qui peut perturber les algorithmes.

PETS 2007

La séquence PETS 2007, http://pets2007.net/, a été filmée dans un aéroport. Comme pour la version 2006, plusieurs vues sont disponibles et nous avons retenu la vue 3 prise en forte plongée. Ici, les effets de la perspective sont plus marqués. Les piétons sont pris de haut et souvent penchés. Contrairement à PETS 2006 dont l'éclairage ambiant fait ressortir les couleurs sombres des vêtements des piétons, ici l'éclairage est assez sombre, rendant les couleurs assez uniformes.

La figure 3.9 présente une image provenant de chacune des séquences PETS.

(a) PETS 2006, vue 4 (b) PETS 2007, vue 3

FIGURE 3.9 – Exemples d'images provenant des séquences PETS

ViCoMo

Issues du projet européen ITEA2 ViCoMo, http://www.vicomo.org/, ces vidéos ont été filmées à l'aéroport d'Eindhoven. Contrairement aux séquences PETS assez courtes, celles-ci durent environ 45 minutes chacune. Nous avons choisi de travailler sur les vues 1 et 2. La vue 1 a un point de vue assez classique tandis que la vue 2 a une forte perspective. Ces deux séquences sont difficiles à traiter car elles contiennent des structures ressemblant à des piétons (comme les mannequins d'exposition à côté des magasins de vêtements) ainsi que de nombreuses occultations. D'autre part les couleurs de l'image tirent sur le gris et il n'est pas rare que la soustraction de fond ait des difficultés à correctement segmenter les objets mobiles qui ressemblent trop au fond. La figure 3.10 présente une image provenant de chacune des séquences ViCoMo.

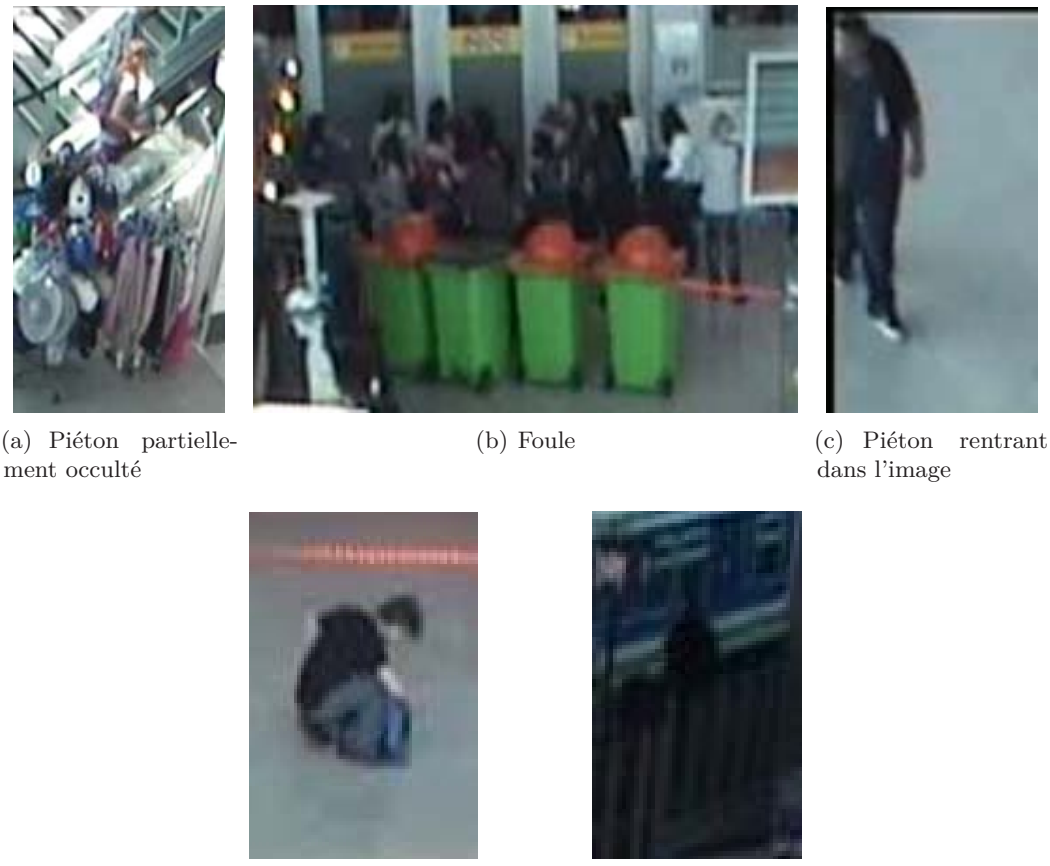
(a) ViCoMo, vue 1 (b) ViCoMo, vue 2

FIGURE 3.10 – Exemples d'images provenant des séquences ViCoMo

3.3.3 Réflexions sur la création de la vérité terrain

Les séquences vidéo présentées précédemment ne sont pas annotées publiquement pour de la détection de piétons, il a donc fallu le faire. Outre le fait que la création d'une vérité terrain est longue et fastidieuse (de l'ordre de 2 à 4 secondes par images en moyenne suivant les vidéos et les outils utilisés), le principal problème est de définir des critères objectifs. En effet, la position et la taille d'un piéton dans une image ne sont pas très nettes, ce qui fait que deux opérateurs différents ne vont pas définir les mêmes boîtes autour des personnes. De plus l'opérateur peut se poser de nombreuses questions. Faut-il annoter tous les piétons, même ceux qu'un humain ne voit pas et ne devine que grâce à leur mouvement ? Un piéton accroupi doit-il être indiqué sachant que le détecteur n'est censé fonctionner que sur des piétons debout ? Si non à partir de quel moment une personne est-elle considérée accroupie et non debout ? Enfin, quand un piéton est-il considéré comme faisant partie de la scène : suffit-il qu'il soit partiellement visible par la caméra ou faut-il attendre qu'il le soit complètement ?

La figure 3.11 présente quelques situations pour lesquelles la vérité terrain est difficile à réaliser de manière objective.



(a) Piéton partiellement occulté

(b) Foule

(c) Piéton rentrant dans l'image

(d) Piéton dans une mauvaise position

(e) Piéton petit, confondu avec le fond et partiellement occulté

FIGURE 3.11 – Exemples de situations ambigües lors de la création de la vérité terrain.

Évaluer une technologie immature ne requiert pas forcément des vérités terrain précises. Cela n'est cependant plus le cas dans le domaine de la détection de piétons. Dès lors, la réponse apportée à toutes ces questions a une grande importance. Pour ne pas sanctionner un détecteur plus sensible que la moyenne qui détecterait les piétons même tronqués, nous avons considéré qu'une personne apparaît dans la scène dès qu'un morceau représentant environ 50% de celle-ci est dans le champ de la caméra.

Certains auteurs (par exemple [Kasturi *et al.*, 2009]) suggèrent, pour réduire la variance des résultats due à l'incertitude sur la vérité terrain, de faire réaliser plusieurs annotations par différents opérateurs. Faute de moyens, celles utilisées dans ce mémoire n'ont été réalisées que par une seule personne.

3.4 Travaux préliminaires justifiant la contextualisation

Contextualiser un détecteur consiste à tenir compte des spécificités de la scène. La collecte d'informations en provenance de celle-ci est une étape essentielle pour connaître l'environnement dans lequel évoluent les piétons à détecter. Plusieurs points sont très importants.

Premièrement, pour reconnaître la forme des piétons, il est primordial de capter au préalable leur apparence générale. Deuxièmement, la structure de la scène est aussi intéressante. Certaines zones peuvent être vides et ne jamais contenir de piétons (murs, périmètres interdits...). Dans d'autres, les déformations dues à la perspective sont importantes. Les piétons y sont par exemple penchés ou petits. Enfin, il est rare que le fond d'une image soit homogène dans tout l'espace, accentuant les disparités d'un point à l'autre. Être capable de repérer tous ces types de régions peut permettre d'y appliquer le traitement adéquat. Par exemple, pourquoi chercher un piéton dans une zone qui n'en contient probablement pas ? De même, pourquoi chercher un piéton droit vu de face alors qu'à cet endroit il est probablement penché et vu de haut ?

Il a été mentionné plusieurs fois que les algorithmes de reconnaissance de formes fonctionnent mieux si la base d'apprentissage et la base de test sont échantillonnées suivant la même loi de probabilité (figure 3.12). Le but va maintenant être de quantifier les progrès liés à la contextualisation d'un détecteur par rapport à un détecteur générique, lorsqu'ils sont employés sur une séquence qui ne partage pas les mêmes caractéristiques que la base d'entraînement générique.

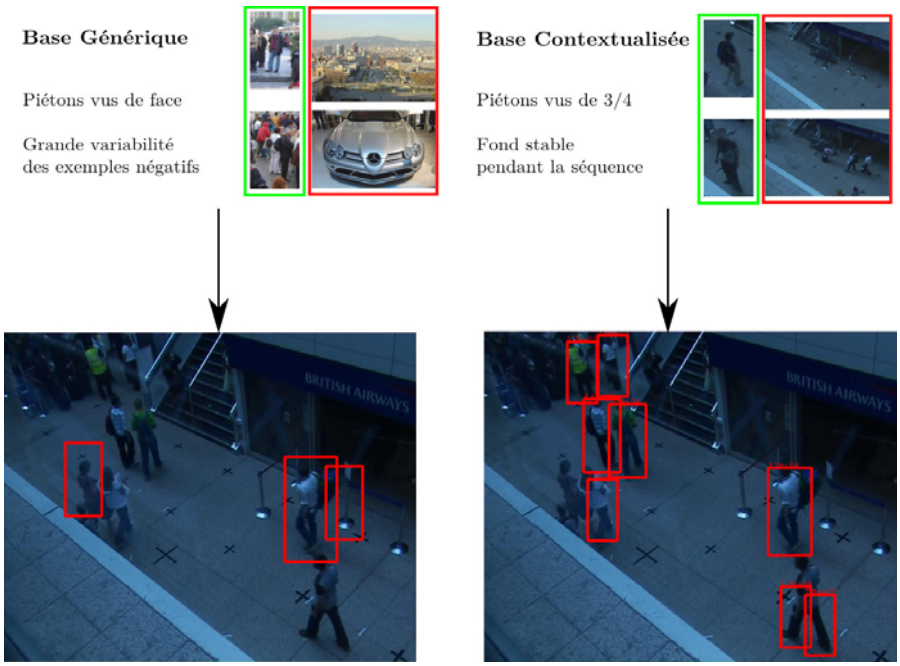


FIGURE 3.12 – Un classifieur ayant été entraîné avec une base générique (image de gauche) donne de moins bons résultats qu'un classifieur contextualisé (image de droite) lorsque le point de vue de l'ensemble d'apprentissage et celui de la détection diffèrent trop.

Obtenir, en vue de la contextualisation, une base d'apprentissage contenant à la fois des exemples positifs et des exemples négatifs provenant de la scène est une tâche ardue. Il est relativement facile d'obtenir les exemples négatifs. Une méthode simple pour y parvenir consiste à ne sélectionner que des portions rectangulaires de l'image à des endroits où la soustraction de fond n'a rien détecté. Un piéton étant la plupart du temps mobile et une image étant le plus souvent constituée de fond, cette approche devrait statistiquement donner des résultats satisfaisants. C'est d'ailleurs l'approche retenue par [Roth *et al.*, 2009] dont le système ne met à jour que la distribution des exemples négatifs.

À l'inverse, construire une base d'exemples positifs repose à un moment ou à un autre sur un détecteur de piétons puisqu'il faut être certain de n'incorporer que des exemples corrects. Or il est difficile avec un détecteur de piétons imparfait de constituer une base d'exemples diversifiée et sans erreurs. Si seuls les exemples négatifs provenant de la scène suffisaient, cela pourrait grandement simplifier le problème de la contextualisation du détecteur.

L'expérience suivante a été réalisée pour connaître l'apport des exemples de la scène dans l'apprentissage, qu'ils soient positifs ou négatifs. Tous les classifieurs ont été testés sur la séquence ViCoMo 1. Cette dernière a été annotée sur plusieurs milliers d'images. Les 10 000 premières servent de base d'entraînement. Un creux de 5 000 images est laissé

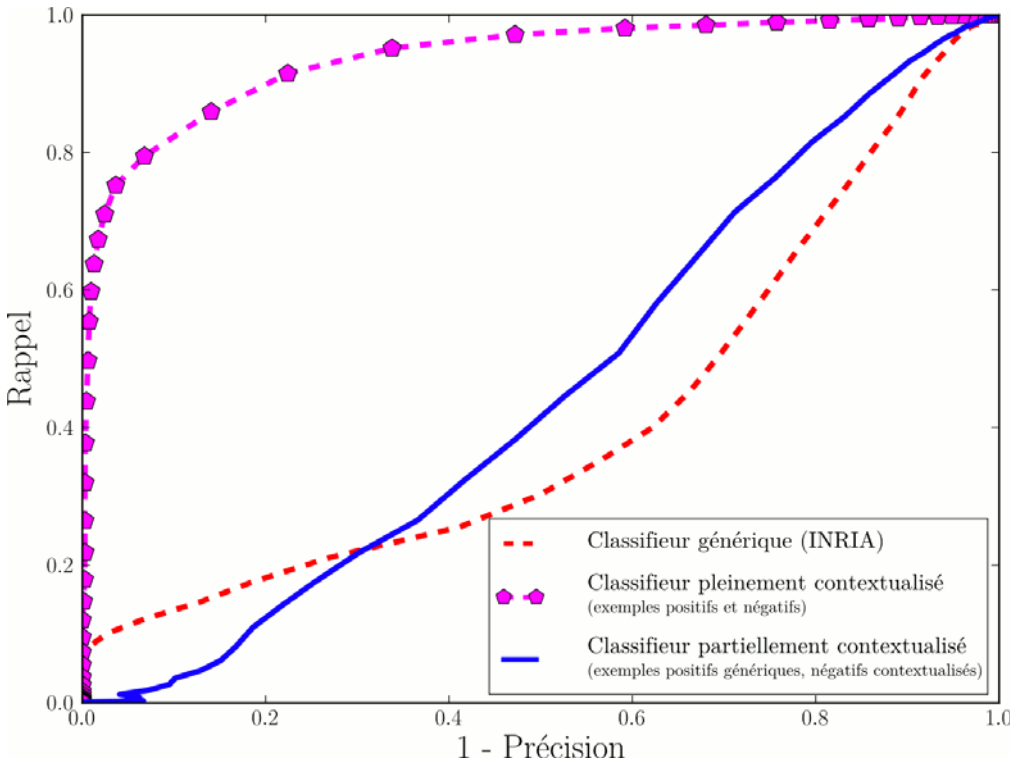


FIGURE 3.13 – Courbes précision-rappel sur la séquence ViCoMo 1 pour les classifieurs générique, partiellement contextualisé (uniquement avec des exemples négatifs) et pleinement contextualisé.

entre la base d'apprentissage et la base de test pour que les piétons présents dans la scène la quittent. Cela limite le risque que les deux bases contiennent des exemples trop similaires et évite de biaiser les résultats. Les images numérotées de 15 000 à 16 000 servent alors de base de validation.

Plusieurs classifieurs ont ensuite été entraînés. Le premier s'appuie sur la base de l'INRIA et est répertorié en tant que classifieur générique. Le second est un classifieur pleinement contextualisé : il apprend exclusivement avec des données issues de la vérité terrain, c'est-à-dire des 10 000 premières images de ViCoMo 1. Il est donc contextualisé avec à la fois des exemples positifs et des exemples négatifs provenant de la scène. Le troisième classifieur est un mélange entre les deux premiers et est partiellement contextualisé. Il utilise la base de positifs de l'INRIA et la base de négatifs de ViCoMo 1.

Les courbes précision-rappel des trois classifieurs sont présentées sur la figure 3.13. Elles indiquent clairement que le classifieur générique offre des performances très inférieures par rapport au classifieur pleinement contextualisé. De plus, le dernier classifieur, contextualisé uniquement à l'aide des exemples négatifs, n'est pas compétitif. Ses performances sont proches de celles du classifieur générique.

Cette expérience montre qu'il est primordial d'entraîner un classifieur contextualisé avec à la fois des exemples positifs et négatifs issus de la scène étudiée. Pour obtenir de bonnes performances, il est donc obligatoire de construire un système capable d'extraire de la séquence des exemples de piétons avec le moins d'erreurs possible.

Maintenant que le but à atteindre est plus précis, nous allons revenir sur les algorithmes d'apprentissage intéressants pour la contextualisation et brosser un panorama des différents systèmes qui ont été proposés dans la littérature.

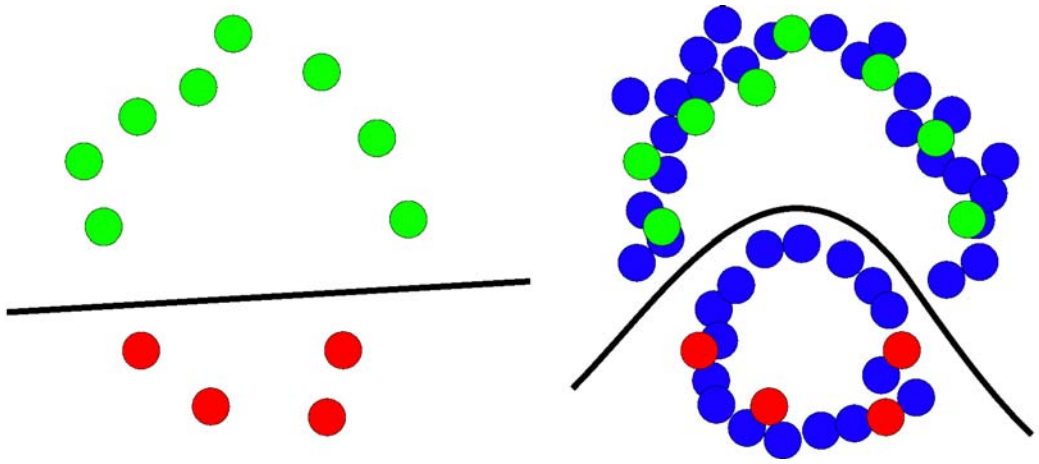
3.5 Apprentissage pour la contextualisation

Les méthodes supervisées ont été étudiées dans le chapitre 2.3.2. Celles-ci sont parfaitement exploitables lorsqu'il existe une base d'apprentissage adaptée au problème. Mais pour notre application, ces données ne sont pas toujours disponibles et le temps de labellisation pour les obtenir est important et rédhibitoire dans un contexte industriel. Par contre, nous disposons d'une grande masse de données puisqu'une fois la caméra installée, le flux vidéo peut être enregistré en continu. Des algorithmes d'apprentissage ont donc été inventés pour tenter d'exploiter cette masse de données non labellisées.

3.5.1 Apprentissage semi-supervisé

L'idée des méthodes semi-supervisées [Chapelle *et al.*, 2006; Zhu, 2008] est d'exploiter à la fois des données labellisées et des données non labellisées. L'ensemble d'apprentissage D est donc scindé en deux. D'un côté D_l contient les données labellisées et de l'autre D_u les observations sans étiquette. L'intérêt étant bien sûr d'avoir le minimum de données étiquetées et le maximum d'observations sans label.

Ces approches semblent très intéressantes dans notre cas. En effet, nous disposons de peu de données labellisées mais d'une grande quantité de vidéos et donc d'un réservoir important d'observations.



(a) Apprentissage supervisé (b) Apprentissage semi-supervisé

FIGURE 3.14 – Principe du fonctionnement de l'apprentissage semi-supervisé. Les exemples sont représentés dans l'espace des observations $\mathbb{X}$. (a) La frontière calculée à l'aide des données labellisées seules (pastilles vertes et rouges) n'est pas représentative une fois les données non labellisées (pastilles bleues) ajoutées (b).

De part sa nature, ce type d'apprentissage est à mi-chemin entre les techniques supervisées étudiées dans la partie 2.3.2 et les approches non supervisées dans lesquelles les observations ne possèdent aucune étiquette, comme par exemple le mean shift (*cf* paragraphe 2.3.3.4), les k-means(*cf* paragraphe 5.3.2.1)... De cette séparation, il ressort deux branches d'algorithmes.

Le clustering semi-supervisé place les observations non labellisées au centre du processus. Elles servent à générer des regroupements entre les données similaires. Le découpage est amélioré avec l'aide des informations apportées par les exemples labellisés. Deux exemples labellisés n'ayant pas la même étiquette ne doivent pas être dans le même groupe. Le problème avec cette approche est qu'il est difficile de contrôler les classes ainsi générées qui peuvent ne pas correspondre à nos attentes.

De l'autre côté, il existe la classification semi-supervisée dans laquelle les exemples labellisés permettent de trouver une frontière qui discrimine les classes d'exemples. Cette séparation est ensuite affinée à l'aide des observations sans étiquette. Une des limites de cette technique est la nécessité d'avoir suffisamment de données labellisées. En détection de piétons, les classes étant déjà définies, nous nous intéressons donc par la suite à ces méthodes.

Les méthodes semi-supervisées ne sont pas magiques. Pour que les observations apportent de l'information sur la loi de distribution des exemples $P(x, y)$ et donc sur le classifieur h, il est nécessaire de faire des hypothèses :

- *Smoothness assumption* : les exemples appartenant à une même classe sont reliés par des zones de forte densité.
- *Cluster assumption* : les frontières entre les classes se trouvent majoritairement dans des zones de faible densité.

– *Manifold assumption* : les exemples d'une même classe, décris dans un espace de grande dimension, se situent dans une variété de dimension inférieure.

Toutes ces hypothèses sont conceptuellement très proches. Les deux premières imposent des contraintes sur la manière de placer la frontière entre les exemples appartenant à des classes différentes. Par exemple sur le schéma 3.14, avec l'ajout de données non-labellisées, la frontière proposée par l'algorithme supervisé n'est plus adéquate car elle sépare des données fortement connectées. La dernière hypothèse permet une atténuation de la « malédiction de la dimension » (*curse of dimensionality* en anglais) qui veut que plus l'espace considéré est grand, plus il est difficile de l'explorer (et donc plus il faut d'exemples d'apprentissage). En restreignant l'algorithme d'apprentissage à des espaces de plus petite dimension, il est possible d'améliorer son pouvoir d'optimisation. Par exemple sur le schéma 3.14, sans les données non-labellisées il n'est pas aisé de savoir que les deux classes sont en fait sur deux cercles différents.

Nous allons maintenant présenter les approches de classification semi-supervisées les plus populaires. Ces méthodes reposent souvent sur un algorithme supervisé qui nécessite un ensemble d'apprentissage complètement labellisé. Il est alors nécessaire d'estimer les étiquettes des données non-labellisées pour pouvoir les injecter dans la base complète. Cette phase fait appel aux hypothèses évoquées précédemment. Le rôle d'étiquetage est dévolu à une entité souvent externe au classifieur.

Le **self-learning** [Rosenberg *et al.*, 2005] consiste à utiliser la réponse du classifieur, entraîné à l'aide des données étiquetées, pour labelliser un exemple. Les exemples pour lesquels le classifieur a une grande confiance sont ajoutés à la base d'apprentissage et un nouveau classifieur est fabriqué. Cette méthode est assez peu robuste car elle souffre d'un défaut de dérive. En effet, les exemples mal labellisés perturbent les réponses pour les exemples suivants et les erreurs se propagent. De plus si le seuil de confiance du classifieur est trop faible, de nombreux faux positifs se retrouvent dans la base. À l'inverse avec un seuil trop haut, seuls les exemples parfaitement reconnus et donc apportant peu d'informations sont retenus.

Le **cotraining** introduit par [Blum et Mitchell, 1998] est un formalisme dans lequel deux classifieurs sont entraînés en utilisant pour chacun des sous-parties indépendantes de la base de données. Par exemple dans [Levin *et al.*, 2003], les auteurs entraînent deux classifieurs, l'un sur le signal d'apparence et l'autre sur celui de la soustraction de fond. Cet algorithme repose sur le fait que les classifieurs entraînés sur différentes parties des données doivent décerner le même label à une observation. Si un des classifieurs répond avec une confiance suffisamment élevée mais que le deuxième n'est pas sûr, alors l'observation est ajoutée dans la base de ce dernier. Pendant l'entraînement les classifieurs doivent s'auto-améliorer. À la fin de la phase d'apprentissage, plusieurs classifieurs performants sont disponibles. Même si le fait d'utiliser deux classifieurs indépendants a pour but d'augmenter la cohérence des prédictions et donc de limiter la dérive des classifieurs, les observations sont toujours labellisées directement grâce à leurs sorties. Le phénomène de dérive n'est donc pas exclu, les données étant rarement totalement indépendantes. D'autre part, l'entraînement de deux classifieurs au lieu d'un seul augmente les temps de calcul. [Javed *et al.*, 2005] ont proposé une version online du cotraining.

Enfin, il existe les **approches avec oracle**. Ce dernier est une entité externe au classifieur qui est chargée de labelliser des exemples non étiquetés avant de les ajouter dans la base. Le classifieur final et l'oracle ne sont pas corrélés, éliminant le principal risque de dérive. De nombreuses méthodes peuvent être expliquées à l'aide du formalisme de l'oracle. Par exemple, [Mallapragada *et al.*, 2008] utilisent une matrice de similarité entre les exemples pour prédire des pseudo-labels. Cette approche n'est pas suffisante pour notre application car nous cherchons à la fois les labels des exemples mais aussi et surtout les observations provenant de notre vidéo. Il est en effet impossible d'apprendre tous les exemples présents dans une vidéo.

Les méthodes intégrant des informations de contexte dans le détecteur et basées sur des oracles sont très populaires. De nombreux oracles différents, de plus en plus complexes, ont été proposés dans la littérature.

3.5.2 Approches avec oracles

Les approches basées sur des oracles présentent de nombreux avantages par rapport à d'autres méthodes semi-supervisées. Elles sont plus robustes que le self-learning dont le risque de dérive est très important et assez simples à mettre en place. De plus elles sont très modulaires puisque, contrairement au cotraining, l'oracle fonctionne seul. La réponse de l'oracle n'est pas liée à celle du classifieur contextualisé, ce qui permet d'intégrer facilement un oracle dans des architectures logicielles déjà existantes et complexes.

Les propriétés d'un bon oracle sont discutées dans le chapitre 4.2 (page 70). D'une manière générale, un oracle ne doit pas faire d'erreur sur le label des exemples étiquetés. La précision de l'oracle détermine en grande partie les performances du système final. Par contre, il n'a pas besoin de détecter tous les piétons de la scène et son rappel peut donc être relativement faible.

De nombreux oracles ont été conçus pour répondre au problème de la contextualisation. Chacun a des caractéristiques différentes en fonction des besoins demandés. Néanmoins de nombreuses similarités existent entre toutes les approches. Les briques de base sont souvent similaires : détecteur de piétons, soustraction de fond, flot optique, algorithmes d'apprentissage... Seules les interactions entre ces différents composants et la manière de les employer changent.

Même s'ils ne conviennent pas à notre application, les oracles les plus simples à mettre en œuvre sont basés sur le travail d'un humain. Dans [Nguyen *et al.*, 2007], une personne corrige les erreurs du classifieur à chaque image. Les nouveaux exemples ainsi labellisés permettent de mettre à jour le classifieur online. Même si cette technique paraît relativement simple à mettre en place, elle pose plusieurs difficultés dans notre cadre applicatif. Outre le fait de reposer sur le travail d'un opérateur, les algorithmes online sont assez sensibles au type d'exemples fournis. Leur donner trop de positifs à la suite a tendance à augmenter le nombre de détections et donc le taux de faux positifs. Le même phénomène apparaît pour les négatifs. De plus, il est probable que le classifieur ne puisse pas atteindre une précision et un rappel de 100%. Ne connaissant pas la borne supérieure atteignable, l'opérateur va essayer d'améliorer en permanence le classifieur au risque de dégrader les performances. Enfin le seuil de détection du classifieur, qui est un paramètre important, est totalement négligé par ce processus. Pour conclure, lors de nos expérimentations nous avons observé qu'il n'est pas si aisé de régler manuellement de

tels classifieurs de manière optimale.

Historiquement, les premiers oracles automatiques étaient basés presque exclusivement sur un signal de mouvement. L'oracle le plus ancien présenté dans cet état de l'art est [Nair et Clark, 2004]. Il est uniquement construit à l'aide d'un algorithme de soustraction de fond. Or comme nous l'avons vu dans le chapitre 2.2.1, un tel algorithme n'est pas robuste. Plusieurs heuristiques sont donc nécessaires pour déterminer si ce qui est désigné par le signal de soustraction de fond est suffisamment fiable pour être appris. Par exemple, si la forme des objets n'est pas comparable à celle d'un humain (mauvaise segmentation) ou si le pourcentage de soustraction de fond dans l'image est trop important (changement d'illumination dans la scène) cela signifie probablement que la soustraction de fond se trompe. Reposant sur un seul signal, cet oracle est par conséquent assez fragile et peut commettre de nombreuses erreurs.

De façon similaire, [Roth *et al.*, 2005] proposent d'entraîner un classifieur de manière online avec uniquement des données mobiles prises dans la scène.

[Celik *et al.*, 2008] ont bâti un oracle non supervisé en utilisant la segmentation de la soustraction de fond. En regroupant les détections de soustraction de fond qui ont une apparence similaire, [Celik *et al.*, 2009] ont étendu l'approche à de la classification multiclasse. Ils sont par exemple capables de différencier plusieurs types d'objets comme les piétons et les voitures. Ils peuvent alors créer un classifieur multiclasse adapté à la scène. L'une des limites de cette méthode est l'utilisation presque exclusive de la soustraction de fond, qui est un signal fournissant peu de sémantique et assez bruité. Comme pour [Nair et Clark, 2004], l'oracle risque de commettre de nombreuses erreurs lors de la construction de la base.

La figure 3.15 présente l'architecture de l'oracle proposé par [Celik *et al.*, 2009] dans le cas où plusieurs types d'objets (piétons, véhicules...) sont présents dans la scène observée.

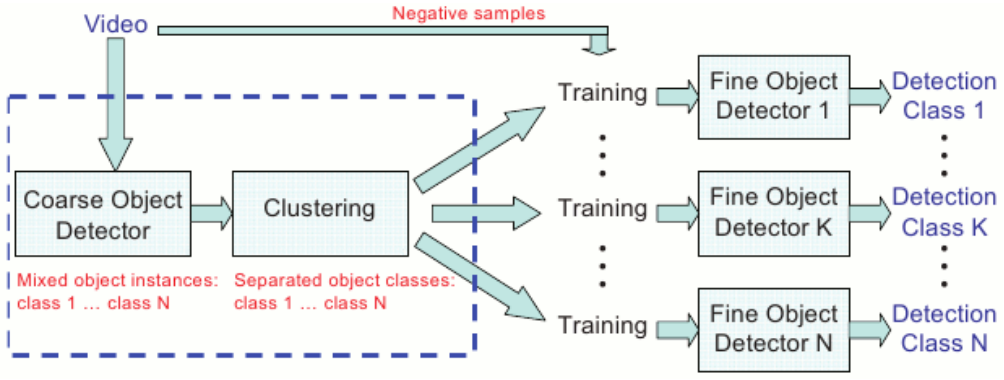


FIGURE 3.15 – Architecture générale de l'oracle proposé par Celik *et al.* (extrait de [Celik *et al.*, 2009])

L'idée sous-jacente de l'approche de [Wu et Nevatia, 2007b] est d'adapter de manière incrémentale un classifieur entraîné sur une base générique à l'apparence des piétons dans la scène. La figure 3.16 présente l'architecture générale du système. L'oracle ici n'est constitué que d'un détecteur par parties basé sur l'apparence. Il n'y a pas de soustraction

de fond pour le compléter. À un exemple est associé un score de détection global et un score pour chaque partie. Un score de détection global faible indique un exemple proche de la frontière de décision et donc un candidat potentiel à prendre en compte pour la mise à jour du classifieur. Si les scores par parties sont suffisamment élevés alors le système a confiance dans cet exemple et l'inclut dans la base. Une fois la base remplie, les détecteurs (les classifieurs par parties et complet) sont modifiés de manière incrémentale pour prendre en compte ces nouvelles informations. Un classifieur fort du boosting est légèrement modifié à chaque itération en ne réentraînant que des classifieurs faibles spatialement proches de ceux sélectionnés lors de l'apprentissage générique. L'idée est, un peu à la manière de l'apprentissage par transfert (*cf* paragraphe 3.5.4.2), que les composantes sélectionnées par le classifieur générique ne sont pas très éloignées de celles qui sont optimales pour le classifieur contextualisé. Ce système ne dispose pas de contrôle extérieur au détecteur pour stabiliser l'ensemble. Son fonctionnement est donc très similaire au *self-learning* et souffre probablement des mêmes défauts : le risque de dérive est important et le système doit déjà être relativement performant dès le départ pour pouvoir amorcer le processus. Or il n'est pas évident que la détection par parties d'un individu soit plus simple à réaliser et soit moins bruitée que la détection globale.

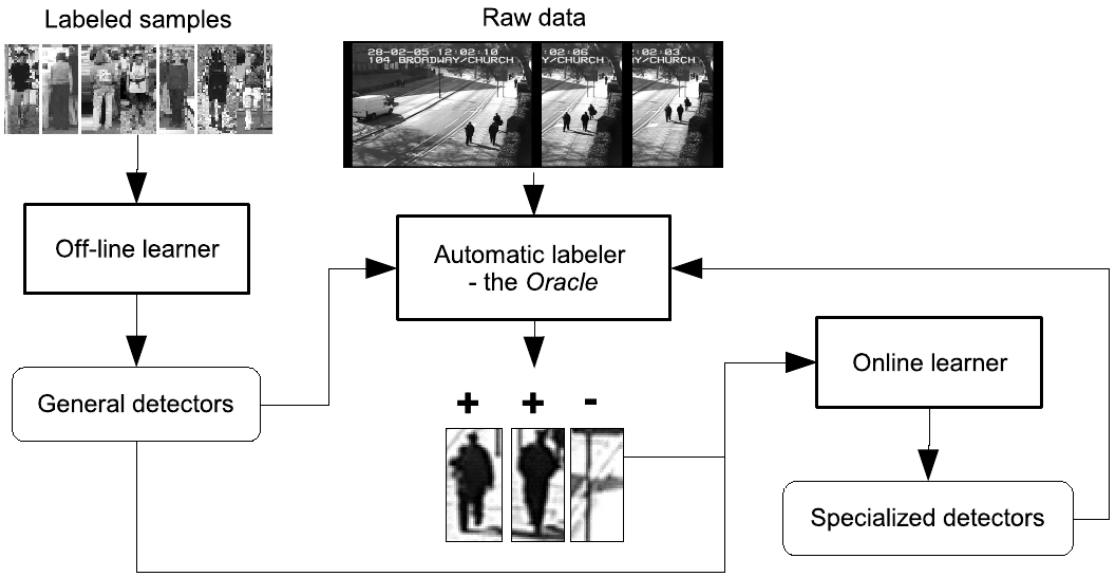


FIGURE 3.16 – Architecture générale de l'oracle proposé par Wu. (extrait de la thèse de B. Wu)

Plus récemment sont apparus des systèmes qui fusionnent plusieurs signaux pour ajouter de la robustesse. [Stalder *et al.*, 2009b] étendent les *classifier grids* et mettent à jour pour chaque classifieur les distributions des exemples négatifs mais aussi des positifs. Concrètement, un détecteur générique collecte les positions des piétons dans l'image. Celles-ci sont ensuite vérifiées par le signal de soutraction de fond. Afin d'augmenter le rappel de l'oracle, [Stalder *et al.*, 2009b] ajoutent un module de tracking. Concrètement, une piste est initialisée sur une détection. Si au bout de quelques images cette piste rencontre à nouveau une détection, alors tous les exemples intermédiaires sont intégrés dans

le classifieur. Le rôle du tracking est d'assurer la continuité spatio-temporelle entre les détections afin d'incorporer des exemples qui n'ont pas été détectés précédemment. Cela permet, contrairement à l'oracle de Wu, de trouver des exemples difficiles pour le classifieur initial et donc importants pour la base contextualisée. Par contre, les algorithmes de tracking ne produisent pas des sorties parfaites et il est assez facile d'incorporer des mauvais exemples, comme des exemples du fond ou des exemples mal centrés, dans la base des piétons. Pour finir, [Stalder *et al.*, 2009b] proposent de réaliser un apprentissage par cotraining entre deux caméras qui observent la même scène de deux points de vue différents. En supposant les deux caméras calibrées entre elles, il est possible d'intégrer les exemples détectés par une caméra dans le classifieur de la deuxième. Cet oracle semble relativement coûteux en temps de calcul.

Enfin, d'une manière générale, pour que l'emploi de classifieurs appris en-ligne soit intéressant, il faut pouvoir les mettre à jour en permanence. L'oracle doit donc fonctionner en parallèle avec le détecteur ce qui augmente les temps de détection.

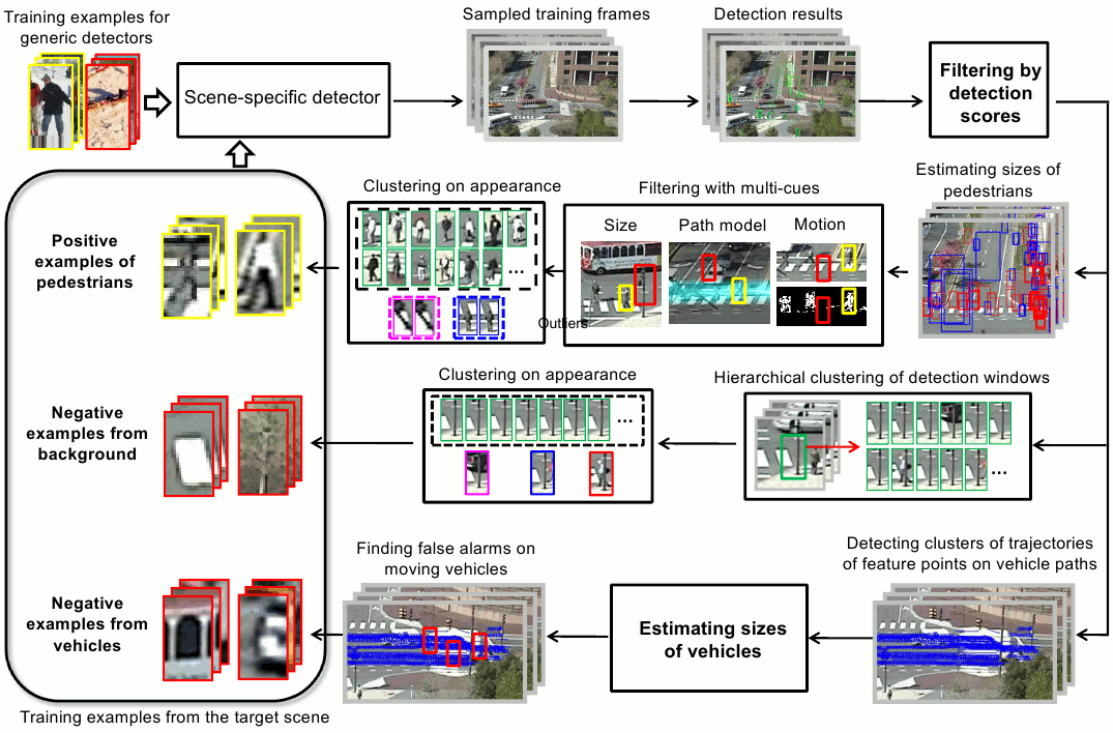


FIGURE 3.17 – Architecture générale de l'oracle proposé par Wang. (extrait de [Wang et Wang, 2011])

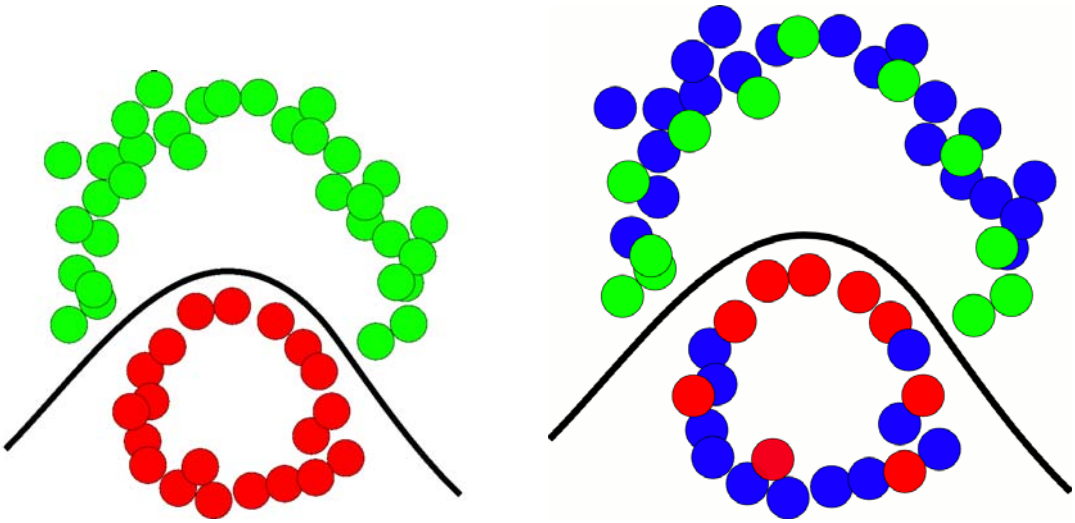
Dernièrement, [Wang et Wang, 2011] ont proposé un nouvel oracle. Il est composé de plusieurs composants. En premier lieu figure un classifieur générique 2D basé sur l'apparence qui analyse plusieurs images. Les détections obtenues passent ensuite par différents filtres avant d'être intégrées dans la base contextualisée. Elles sont vérifiées et mises en correspondance avec les sorties des algorithmes de soustraction de fond et de mouvement, puis regroupées en clusters selon leur apparence. Seuls les exemples se retrouvant dans le cluster majoritaire sont conservés. Dans une scène fixe, les faux positifs

se retrouvent souvent au même endroit sur plusieurs images consécutives. En exploitant ce phénomène, il est possible de déduire les faux positifs les plus probables et de les ajouter dans la base des négatifs. Là encore un regroupement est effectué pour ne garder que les exemples les plus sûrs. Un nouveau classifieur est alors entraîné avec les exemples de la base générique et de la base contextualisée et le processus recommence pendant environ 5 à 10 itérations. Au fur et à mesure de l'avancement de la contextualisation, le système déduit la taille approximative des piétons dans la scène permettant ainsi de supprimer les boîtes détectées qui sont inadéquates. La figure 3.17 présente l'architecture générale du système proposé par [Wang et Wang, 2011].

Dans [Wang *et al.*, 2012], les auteurs améliorent leur système en y incorporant de l'apprentissage par transfert sur les exemples (*cf* paragraphe 3.5.4.1). Ainsi seuls les exemples de la base générique qui sont proches de ceux de la scène sont utilisés dans l'apprentissage contextualisé.

3.5.3 Apprentissage actif

L'apprentissage actif [Settles, 2009; Bondu et Lemaire, 2007] est une autre catégorie d'algorithmes exploitant des données non labellisées. Ce sont des méthodes interactives dans lesquelles l'apprenant (c'est-à-dire l'algorithme d'apprentissage) peut influer sur le choix des exemples d'apprentissage. L'intérêt repose sur le fait que l'algorithme est le mieux placé pour savoir quels sont les exemples qui, s'ils étaient labellisés, pourraient apporter le plus d'informations par rapport à ceux déjà étiquetés. Le but étant de réduire le nombre d'exemples nécessaires pour l'entraînement d'un classifieur (voir la figure 3.18).



(a) Apprentissage supervisé (b) Apprentissage actif

FIGURE 3.18 – Illustration de l'intérêt de l'apprentissage actif. (a) Avec un apprentissage supervisé, tous les exemples doivent être étiquetés pour trouver la bonne séparation entre les classes. (b) Avec un apprentissage actif, seuls les exemples les plus pertinents (ici ceux qui sont proches de la frontière) ont été labellisés.

Comme pour l'apprentissage semi-supervisé, de nombreuses stratégies ont été proposées dans la littérature. Deux grandes familles d'approches existent :

- les approches constructives dans lesquelles des exemples sont construits de toute pièce pour que ceux-ci soient les plus pertinents possible ;
- les approches sélectives dans lesquelles les exemples pertinents sont choisis dans un ensemble donné.

Les stratégies sélectives (ou échantillonnage sélectif) sont les plus populaires puisqu'il est souvent plus facile de collecter des exemples que d'en fabriquer de toute pièce. Elles sont résumées dans l'algorithme 4. Concrètement une fonction d'utilité note la pertinence d'ajouter un nouvel exemple par rapport à un classifieur.

$$Utile : \mathbb{X} \times \mathbb{H} \longrightarrow \mathbb{R} \tag{3.5}$$

avec $\mathbb{X}$ l'espace des observations et $\mathbb{H}$ l'ensemble des classifieurs.

À chaque itération, l'algorithme sélectionne donc l'exemple le plus pertinent, demande à l'oracle de le labelliser et met à jour le classifieur avec ce nouvel élément.

Algorithme 4: Apprentissage actif par échantillonnage sélectif. (extrait de [Bondu et Lemaire, 2007; Muslea, 2002])

Entrées :
Un classifieur $h \in \mathbb{H}$ et un algorithme d'apprentissage A,
Un ensemble d'apprentissage : $L \cup U$, les ensembles d'exemples labellisés et non labellisés,
Une fonction d'utilité, $Utile : \mathbb{X} \times \mathbb{H} \longrightarrow \mathbb{R}$ qui estime l'apport d'un exemple par rapport à un modèle,
Le nombre N d'exemples à étiqueter

Sorties :
Un classifieur h

Algorithme :
tant que *nombre d'itérations* $< N$ **faire**
(1) Entraîner le classifieur h grâce à A, L et éventuellement U,
(2) Rechercher l'exemple $\hat{x} = \arg\max_{x \in U} Utile(x, h)$,
(3) Demander l'étiquette $y_{\hat{x}}$ à l'oracle,
(4) Supprimer $\hat{x}$ de l'ensemble U et ajouter $(\hat{x}, y_{\hat{x}})$ à L.
fin

À partir de ce cadre, il est possible de définir plusieurs algorithmes de sélection active. L'échantillonnage par incertitude [Lewis et Gale, 1994] vise à choisir les exemples pour lesquels le classifieur est le moins confiant. L'échantillonnage par comité de modèles [Seung *et al.*, 1992] consiste à entraîner plusieurs classifieurs sur les données labellisées, chacun représentant une hypothèse différente. L'idée est de sélectionner les exemples pour lesquels ces classifieurs ne sont pas d'accord entre eux. Au fil du temps, l'ajout d'exemples dans la

base peut empêcher certains classifieurs de trouver des modèles cohérents. Les hypothèses correspondant à ces classifieurs ne sont alors plus valides, permettant ainsi de réduire l'espace des classifieurs admissibles.

L'apprentissage actif et l'apprentissage semi-supervisé tentent donc de répondre au même problème mais de manière différente. L'apprentissage semi-supervisé déduit à l'aide d'hypothèses et de la structure des données (labellisées et non labellisées) des informations sur la distribution des exemples et donc sur les labels sous-jacents. Cela revient à exploiter la partie connue des informations contenues dans la base. De son côté l'apprentissage actif a pour but d'explorer les aspects inconnus de la base en faisant étiqueter les exemples les plus pertinents par un intervenant extérieur.

3.5.4 Apprentissage par transfert

Les algorithmes d'apprentissage présentés jusqu'à maintenant, font l'hypothèse que les données d'apprentissage et de test sont dans le même espace $\mathbb{X}$ et partage la même distribution P.

Le *transfer learning* [Pan et Yang, 2010] tente de répondre à la problématique suivante. Soient deux ensembles d'apprentissage : D_s les données génériques (ou sources) et D_t les données de test (ou cibles). Comment exploiter les connaissances contenues dans D_s pour améliorer la classification sur D_t ?

Il existe plusieurs familles de *transfer learning* qui répondent à des besoins différents. Dans le cas de la contextualisation, les domaines de classification (l'espace des observations $\mathbb{X}$ et l'espace des classes $\mathbb{Y}$) sont identiques entre les données génériques et celles issues de la scène étudiée. Néanmoins les distributions des exemples sur les deux domaines sont différentes et l'hypothèse faite par les algorithmes d'apprentissage classiques n'est pas valide. Cette catégorie de problèmes est traitée par les méthodes appelées : *inductive transfer learning*. Leur but est d'inférer un modèle dans le domaine cible avec l'aide des données sources.

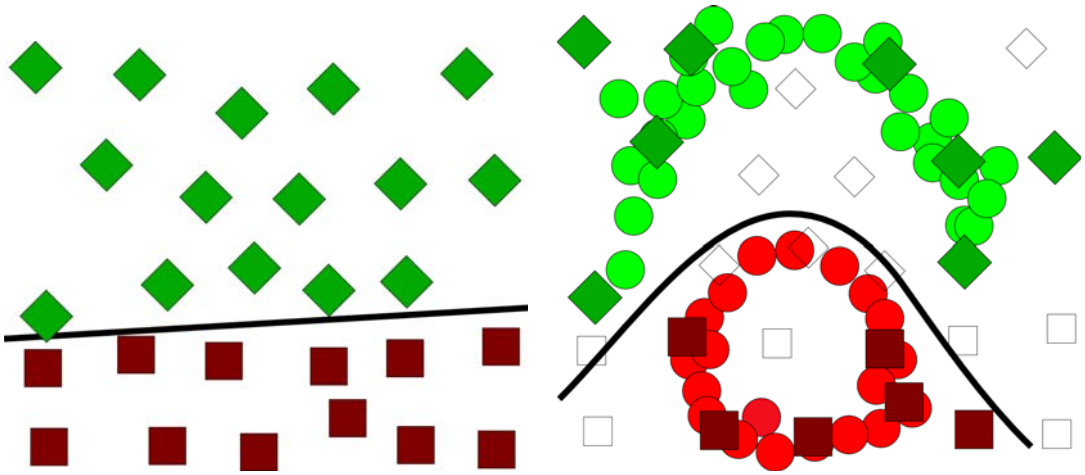
Plusieurs algorithmes existent en fonction du transfert souhaité entre la source et la cible. Les principaux cas possibles sont le transfert sur les exemples ou sur les composantes.

3.5.4.1 Transfert sur les exemples

L'idée exploitée ici est que tous les exemples sources ne sont pas bénéfiques à l'apprentissage cible. Certains, trop éloignés, peuvent avoir une influence négative sur l'estimation de la fonction de prédiction. Il convient donc de les supprimer et dans le même temps d'augmenter l'influence des exemples similaires à ceux de l'ensemble cible (figure 3.19).

[Dai *et al.*, 2007] ont proposé TrAdaboost. Dans cet algorithme de boosting, les exemples sources mal classés voient leurs poids diminuer pendant l'apprentissage. Ainsi dans les itérations suivantes les exemples qui correspondent à la nouvelle distribution ont plus d'influence.

Dans [Wang *et al.*, 2012], les auteurs ont une approche un peu différente. Ils calculent une similarité entre les exemples sources et cibles. Un exemple source qui est souvent proche des exemples cibles est mieux noté que celui qui ne l'est jamais.



(a) Apprentissage supervisé (b) Apprentissage par transfert sur les exemples

FIGURE 3.19 – Illustration de l'intérêt de l'apprentissage par transfert. Les exemples sources, représentés par des carrés et les exemples cibles, représentés par des cercles, ne sont pas distribués de la même façon dans l'espace des descripteurs. (a) Un apprentissage supervisé, portant uniquement sur les données sources ne permet pas de trouver la bonne séparation entre les classes pour les données cibles. (b) Avec un apprentissage par transfert, les exemples sources qui sont trop éloignés des exemples cibles ne sont pas pris en compte. L'algorithme d'apprentissage peut alors estimer la frontière de séparation entre les classes correctement.

3.5.4.2 Transfert sur les composantes

Le transfert de composantes consiste à trouver les composantes du descripteur qui représentent au mieux les exemples des domaines source et cible. Le but est de minimiser les divergences entre les deux domaines.

3.6 Conclusion

Ce chapitre a abordé le problème de la contextualisation, c'est-à-dire de la prise en compte par le détecteur d'informations en provenance de la scène. Il a tout d'abord été montré, au travers de quelques situations concrètes, que l'apprentissage supervisé n'est pas forcément une solution idéale pour construire un classifieur adapté à une situation précise.

Les expériences que nous avons menées dans ce chapitre et qui sont présentées sur la figure 3.13 ont clairement mis en évidence l'importance de la contextualisation et de la prise en compte de l'apparence des piétons de la scène étudiée, dans le processus de construction du classifieur.

Ces nouvelles contraintes imposent l'emploi d'autres algorithmes d'apprentissage pour construire le classifieur. En particulier les algorithmes semi-supervisés, reposant sur le concept d'oracle, permettent grâce à l'ajout d'exemples issus de la scène dans la base d'apprentissage, d'améliorer significativement les performances d'un classifieur.

Ces deux premiers chapitres ont passé en revue les différentes briques qui composent généralement un détecteur de piétons. La littérature est riche et de nombreuses variantes sont proposées à chaque fois.

Ceci conclut l'état de l'art. La partie suivante est consacrée aux travaux réalisés au cours de cette thèse.

Partie II : Travaux réalisés

Positionnement de la thèse

L'enjeu de cette thèse est de proposer un système contextualisé complet de détection de piétons. La plupart des approches présentées dans le chapitre précédent ont comme objectif d'entraîner un classifieur spécialisé sur une scène. Notre but est d'essayer d'aller plus loin en proposant de contextualiser non seulement le classifieur mais aussi les différentes parties qui composent le détecteur de piétons. De nombreux domaines de la contextualisation sont ainsi abordés, parmi lesquels figurent la création automatique de la base d'apprentissage contextualisée à l'aide d'un oracle, la sélection des exemples pertinents issus de cette base ou encore la stratégie de parcours du détecteur dans l'image et le réglage de son seuil. L'apport de la contextualisation a été évalué à chaque étape afin de valider les évolutions du système.

Plusieurs oracles ont été construits et sont présentés dans le chapitre 4 : un premier en 2D qui ne s'appuie sur aucune connaissance *a priori* de la scène et un deuxième en 3D utilisable lorsque la caméra est calibrée. La détection de piétons 3D contrainte par le plan du sol grâce à la calibration est plus précise et donc plus performante que son homologue 2D. Cependant toutes les scènes ne se prêtent pas bien à l'hypothèse selon laquelle les piétons n'évoluent que sur un seul plan. De plus, il est possible que la caméra bouge légèrement au fil du temps et l'obtention des paramètres de la caméra est une phase difficile à automatiser. À noter que différentes approches pour autocalibrer une caméra reposent sur l'utilisation d'un détecteur de piétons 2D dont les performances sont cruciales pour la bonne estimation des paramètres de la caméra. Le premier oracle dont le but est de construire un détecteur 2D performant est donc intéressant dans ces cas là.

Les oracles que nous proposons sont constitués des mêmes briques que celles généralement exploitées par les oracles de la littérature. Par exemple, de nombreux oracles reposent sur un algorithme de soustraction de fond [Nair et Clark, 2004; Celik *et al.*, 2008; Stalder *et al.*, 2009b; Wang et Wang, 2011], sur un algorithme de tracking [Stalder *et al.*, 2009b; Wang et Wang, 2011]. Les différences majeures dans la réalisation d'un oracle proviennent donc de la façon dont tous ces signaux sont utilisés. Par exemple, contrairement aux oracles de la littérature qui s'appuient sur la segmentation fond/forme, dans le cas de notre oracle 2D nous avons testé l'emploi de détecteurs de piétons travaillant avec les signaux de soustraction de fond et de flot optique afin de rendre l'utilisation de ces signaux plus robustes. De même, la fusion des résultats fournis par les composants de l'oracle peut différer. [Wang et Wang, 2011] précisent que les différents signaux sont fusionnés avant le regroupement des boîtes. La fusion de notre oracle 2D a lieu après le regroupement, tandis qu'elle a lieu en même temps pour l'oracle 3D.

Comparer deux oracles et donc mesurer précisément l'apport de ces quelques modifications est très difficile. En effet, il s'agit, comme nous l'avons vu, de systèmes très complexes qui mettent en œuvre de nombreux algorithmes différents. La plus grande difficulté lors de la réalisation d'un oracle est de bien intégrer ces composants et de les régler ensemble (paramètres des algorithmes, stratégies de fusion) afin d'obtenir un système cohérent. Malheureusement, les articles présentant les différents oracles manquent souvent de détails, laissant une part d'interprétation non négligeable au lecteur. Ces raisons expliquent pourquoi nous nous comparons rarement à l'état de l'art, notamment aux approches semi-supervisées utilisant les oracles cités précédemment.

Une fois la base construite à l'aide de notre oracle, il est possible d'entraîner un classifieur contextualisé. Tout comme [Wang et Wang, 2011], nous avons choisi d'employer un classifieur offline. En effet, lors de nos tentatives, nous avons trouvé qu'il est difficile de régler de manière optimale un classifieur online même manuellement. Il n'est pas facile de fournir à un classifieur les bons exemples à chaque image et quelques erreurs suffisent à faire dériver le système, chose qu'il n'est pas aisé de corriger par la suite. Par contre contrairement aux systèmes proposés par [Wu et Nevatia, 2007b; Wang et Wang, 2011], seuls les exemples provenant de la scène sont intégrés dans la base d'apprentissage contextualisée. Ces derniers sont nombreux et cette précaution évite de perturber les distributions des exemples avec des données génériques.

La base contextualisée étant entièrement labellisée par l'oracle, le cadre est légèrement différent de celui de l'apprentissage actif. Néanmoins, il nous a semblé important d'étudier l'impact du choix des exemples lors de la construction du classifieur. Nous avons essayé plusieurs stratégies que nous présentons dans le chapitre 5. Au cours de nos travaux, nous avons observé des variations importantes lors de l'évaluation de classifieurs construits suivant le même processus ainsi qu'une saturation des performances, c'est-à-dire que malgré des choix de paramètres différents aucun classifieur n'arrivait à dépasser un certain degré de performances. Nous supposons que cette saturation est partiellement expliquée par le modèle de représentation des piétons (combinaison du descripteur et de l'algorithme d'apprentissage) choisi. Cela nous a amené à proposer une approche pour le complexifier sans augmenter les temps de calcul lors de la détection. Pour cela nous nous servons de la structure de la scène pour contextualiser le détecteur spatialement. Notre approche se situe à mi-chemin entre celles de [Wu et Nevatia, 2007b; Wang et Wang, 2011] qui entraîne un classifieur unique pour toute l'image et celle de [Stalder *et al.*, 2009b] qui construit un modèle de piéton par point de l'image.

Enfin la description de la plupart des systèmes de la littérature s'arrête une fois le classifieur contextualisé entraîné. Mais dans un cadre applicatif, il est important de s'intéresser à tous les composants du détecteur. Pour gagner du temps lors de la détection, nous proposons un mécanisme simple consistant à explorer les zones où l'oracle a détecté au moins un piéton. Une heuristique similaire existe dans [Stalder *et al.*, 2009b]. D'autre part, la contextualisation permet de construire un classifieur aux performance accrues, mais il est nécessaire de le placer à son point de fonctionnement optimal. Le réglage des parties du détecteur est abordé dans le chapitre 6.

La table 3.1 résume les différences entre les méthodes présentées dans l'état de l'art et notre approche.

TABLE 3.1 – Propriétés des différents systèmes pour la contextualisation d'un détecteur de piétons. (SdF = soustraction de fond, FO = flot optique)

	Apprentissage	**Signal**	**Algorithme semi-supervisé**	**Calibrage**
Systèmes mono-signal				
[Nair et Clark, 2004]	online	SdF	règles sur la SdF	2D
[Celik *et al.*, 2009]	offline	SdF	oracle	2D
[Wu et Nevatia, 2007b]	incrémental, par parties	Apparence	self-learning	2D
Systèmes multi-signaux				
[Stalder *et al.*, 2009b]	online, *classifier grids*	Apparence, SdF, Tracking	oracle, cotraining	3D, multi-caméras
[Wang et Wang, 2011]	offline	Apparence, FO, Tracking	oracle	2D avec calibrage faible
Notre approche	offline, régions, réglage du seuil	Apparence, SdF, FO, Tracking	oracles	2D ou 3D

La contextualisation d'un détecteur est un domaine riche faisant appel à de nombreux concepts différents. La diversité des sujets abordés dans cette thèse a permis des avancées, mais a également soulevé de nouvelles questions auxquelles devront répondre de futurs travaux.

Partie II : Travaux réalisés

4

Contextualisation par oracle

Sommaire

4.1 Introduction . . . 70
4.2 Définition et propriétés d'un oracle . . . 70
4.3 Oracle 2D . . . 71
4.3.1 Caractéristiques de l'oracle 2D . . . 71
4.3.2 Fusion des signaux : intersection . . . 74
4.3.2.1 Génération des exemples positifs . . . 74
4.3.2.2 Génération des exemples négatifs . . . 76
4.3.3 Implémentation . . . 77
4.3.3.1 Implémentation de l'oracle . . . 77
4.3.3.2 Implémentation du classifieur contextualisé . . . 80
4.3.4 Validations expérimentales 2D . . . 82
4.3.4.1 PETS 2006 . . . 82
4.3.4.2 PETS 2007 . . . 84
4.3.4.3 Performances du détecteur contextualisé . . . 86
4.3.5 Améliorations possibles . . . 87
4.3.5.1 Oracle et tracking . . . 87
4.3.5.2 Oracle et seuil adaptatif . . . 92
4.4 Oracle 3D . . . 95
4.4.1 Caractéristiques de l'oracle 3D . . . 95
4.4.2 Fusion des signaux : minimisation d'une fonction . . . 95
4.4.3 Implémentation . . . 97
4.4.4 Validations expérimentales 3D . . . 102
4.4.4.1 PETS 2006 . . . 102
4.4.4.2 ViCoMo 2 . . . 104
4.5 Conclusion . . . 106

4.1 Introduction

La contextualisation d'un détecteur consiste à extraire de l'information provenant de la scène et à l'injecter dans le détecteur de piétons afin d'augmenter ses performances.

Le chapitre 3 a détaillé le principe de la contextualisation ainsi que les différentes approches existantes dans la littérature. Notre choix s'est porté sur les méthodes basées sur des oracles dont le rôle est de constituer une base d'apprentissage contextualisée. Cette dernière sert ensuite à l'entraînement d'un classifieur. L'oracle doit donc collecter une base d'exemples positifs et d'exemples négatifs issus de la vidéo. Cela signifie qu'il doit être capable, tout comme le détecteur final, de résoudre le problème de la reconnaissance de la forme des piétons. Néanmoins ces deux entités ne partagent pas les mêmes buts et diffèrent dans leur réalisation. La définition et les propriétés de l'oracle sont tout d'abord rappelées dans la partie 4.2.

La suite du chapitre détaille les différents oracles que nous avons construits pendant cette thèse. Tout d'abord, nous nous sommes intéressés aux oracles purement 2D (partie 4.3). Leur avantage est qu'ils peuvent être utilisés dans tous les types de situations car ils font très peu de suppositions sur la géométrie de la scène. Ensuite nous nous intéressons aux oracles 3D (partie 4.4) qui ne peuvent être employés que dans le cas où les paramètres de calibration de la caméra sont connus et où la scène est considérée come étant plane.

Une grande partie de ce chapitre est consacrée à l'évaluation des différents oracles proposés. Les séquences utilisées ont toutes été présentées dans le chapitre précédent 3.3.2.

4.2 Définition et propriétés d'un oracle

Pour être utilisable facilement et à grande échelle, la contextualisation doit être automatique. Dans un premier temps, pour capturer l'apparence des personnes dans la scène, il faut extraire de notre vidéo des exemples de piétons et de fond en vue d'entraîner un classifieur. Le chapitre 3 a présenté les méthodes semi-supervisées (*cf* partie 3.5.1, page 53) qui répondent bien à ce problème : elles extraient de l'information d'une grande base de données non étiquetées et de la vidéo. Pour ce faire, nous avons décidé de créer un oracle. L'avantage de cette méthode, par rapport à d'autres méthodes semi-supervisées est sa robustesse et sa flexibilité. L'étiquetage de nouveaux exemples ne reposent pas sur celui des exemples précédents comme dans le cas du self-learning. De plus, un oracle est un système modulaire dans lequel il est possible d'intégrer un large panel d'outils complémentaires.

Le rôle de l'oracle est d'annoter automatiquement une vidéo, c'est-à-dire de trouver des observations de piétons et de fond, ainsi que les labels correspondants. Les couples (observation, label) serviront ensuite à entraîner un classifieur de manière supervisée. En ce sens, l'oracle est bien un détecteur de piéton, mais il ne possède pas les mêmes propriétés que le détecteur final, qui doit non seulement être temps réel, mais aussi détecter le maximum de piétons avec le minimum de faux positifs. En d'autres termes, le détecteur final doit avoir un rappel et une précision élevés.

Le point primordial à respecter pour un oracle est de faire le moins de fautes possible lors de la recherche d'information. Sa précision doit donc être la plus élevée possible pour ne pas bruiter la base d'apprentissage contextualisée. L'oracle peut commettre deux types d'erreurs :

- les erreurs de position : le piéton n'est pas centré correctement,
- les erreurs de label : l'exemple n'est pas bien étiqueté, un piéton est considéré comme du fond ou inversement.

Pour atteindre des taux de précision élevés en toutes circonstances, il est nécessaire de relâcher certaines contraintes. Le but étant de créer une base d'apprentissage, il n'est pas pénalisant pour l'oracle d'omettre certains piétons, du moment qu'à l'issue de son travail, la base en contienne suffisamment. Son rappel peut donc être très faible si la vidéo est assez longue et contient suffisamment de piétons.

Enfin en fonction de l'application finale visée, l'oracle n'est pas nécessairement temps réel, car la mise à jour du classifieur contextualisé n'est pas obligatoirement faite en permanence. La méthode que nous employons comporte deux temps. Dans une première phase, le système apprend. L'oracle peut étudier autant d'images qu'il le souhaite sans impératif de temps de calcul. Dans une deuxième phase, le système est opérationnel. Les piétons sont localisés en temps réel par le détecteur contextualisé.

Puisque la contextualisation du détecteur est une étape automatique, l'oracle doit être générique et donc totalement indépendant de la scène traitée. En particulier, cela impose des contraintes sur les réglages de l'oracle qui ne doivent pas dépendre de la vidéo traitée. Par exemple, les seuils de détection ne doivent pas être changés manuellement en fonction des circonstances. Si tel était le cas, l'oracle commettrait des erreurs dans la base dans les situations délicates et son utilisation nécessiterait une intervention humaine pour faire les corrections adéquates. Cela est bien sûr exclu pour l'application visée.

Table 4.1 – Propriétés importantes d'un oracle et d'un détecteur de piétons.

Détecteur	Rappel	Précision	Temps réel	Remarques
Générique	±	±	oui	sous-optimal
Oracle	±	++	non	indépendant de la scène
Contextualisé	+	+	oui	spécifique à une scène

La table 4.1 résume et compare les principales propriétés des différents détecteurs de piétons rencontrés dans ce travail. Les différentes approches, que nous proposons pour fabriquer des oracles répondant à ce cahier des charges, sont détaillées dans la suite de ce chapitre.

4.3 Oracle 2D

4.3.1 Caractéristiques de l'oracle 2D

Après avoir étudié les propriétés souhaitées de l'oracle, nous avons commencé notre travail en construisant un oracle 2D. L'avantage d'un tel oracle est qu'il s'adapte à

n'importe quelle situation, sans travail supplémentaire de la part d'un opérateur. En effet, en supposant que la caméra est installée de manière standard (pointée en légère plongée, pas d'objectif grand angle), le processus de contextualisation peut démarrer immédiatement. En particulier aucune étape de calibration de la caméra n'est requise. Le schéma 4.1 présente les principales étapes de l'oracle 2D intégré dans la chaîne de contextualisation globale. Le schéma 4.6 (page 81) détaille l'architecture complète de l'oracle 2D.

La première étape pour l'oracle est de trouver des exemples positifs. La scène n'étant pas connue *a priori*, nous avons logiquement décidé d'utiliser des méthodes temporelles comme la soustraction de fond et le flot optique (*cf* partie 2.2, page 14) qui sont très adaptables. Cependant, ces approches détectent tous les objets mobiles. Pour augmenter la probabilité que les exemples choisis soient effectivement des piétons, un module de reconnaissance de formes basé sur l'apparence est nécessaire. Une étape de fusion entre ces signaux est ensuite réalisée et assure que seuls les piétons sont extraits de la vidéo.

Il existe plusieurs façons de combiner ces méthodes et de fusionner tous les signaux obtenus. Par exemple, les segmentations de l'image produites par la soustraction de fond et le flot optique pourraient être employées directement. Toutes les « taches » découvertes seraient ensuite validées par le signal d'apparence. Cette approche est assez peu coûteuse en terme de temps de calcul. Néanmoins, les segmentations résultantes des algorithmes de soustraction de fond et de flot optique peuvent être très bruitées (objets partiellement détectés, surdétections dues aux conditions d'éclairage ou au bruit de l'image). La création d'objets non pertinents, qui sont potentiellement difficiles à filtrer, risquerait de déclencher la création de faux positifs par l'oracle. Pour limiter les conséquences du bruit dans les segmentations, il a semblé plus favorable de privilégier le classifieur du signal d'apparence, et de laisser légèrement en retrait les signaux temporels. Les résultats obtenus sur la séquence PETS 2006 (*cf* figure page 83) confirment que le signal d'apparence peut être plus performant que les signaux temporels. Néanmoins, des disparités existent en fonction des vidéos comme le montrent les expériences sur la séquence PETS 2007 (*cf* figure page 85).

Initialement nous avons choisi d'exploiter les signaux temporels à l'aide du formalisme des fenêtres glissantes basé sur un parcours en 2D (*cf* paragraphe 2.3.3.3, page 35). Au lieu d'extraire les objets mobiles de la scène, nous nous contentons de regarder les zones avec une forte densité de pixels étiquetés comme objet. Pour cela, nous évaluons le ratio entre le nombre de pixels allumés et le nombre de pixels de la boîte. Cela permet de filtrer les petits objets (piétons incomplets) ainsi qu'une partie du bruit quand la densité de pixels labellisés comme objet à l'échelle d'une boîte est trop faible. En outre, même si ces objets sont des personnes, leur taille est trop restreinte, ils ne sont pas adaptés aux réglages du futur détecteur contextualisé et ils risqueraient de pénaliser l'apprentissage contextualisé. Ils doivent donc être éliminés. Comme dans le cadre de la détection classique, une étape de regroupement (*cf* paragraphe 2.3.3.4, page 37), consistant à maximiser la fonction de densité des détections, est nécessaire pour agglomérer les boîtes validées.

Cette approche préliminaire soulève néanmoins deux problèmes. Tout d'abord, elle n'exploite qu'une information binaire fond/objet et non toute la richesse de la soustraction de fond ou du flot optique. Par exemple dans le cas de la soustraction de fond, la distance

du pixel courant au modèle du fond à cet endroit, fournit une mesure de l'incertitude de la classification en ce point. Le second problème est plus important. La densité des pixels allumés dans une boîte n'est pas une bonne fonction à maximiser car une petite boîte centrée sur le thorax d'un piéton (donc avec beaucoup de pixels objets) aura plus de poids qu'une plus grande englobant tout le piéton (mais avec des pixels fond autour). Ce problème apparaît ici car la détection 2D ne pose pas de contraintes sur la taille des piétons.

Pour pallier ces deux défauts, nous avons décidé de construire trois classifieurs, un pour chaque signal : apparence, soustraction de fond et flot optique. Le classifieur d'apparence fournit de l'information sur la nature et la position des objets détectés. En ce qui concerne les deux signaux temporels, le simple comptage des pixels labellisés est trop fragile. La construction d'un classifieur à l'aide d'un algorithme d'apprentissage permet de construire un modèle plus robuste au bruit. Il peut paraître contradictoire d'utiliser un algorithme d'apprentissage pour construire des classifieurs de soustraction de fond et de flot optique, puisque cela rend ces méthodes moins adaptables. Néanmoins, en pratique cette approche, bien que coûteuse en temps de calcul, donne des résultats intéressants.

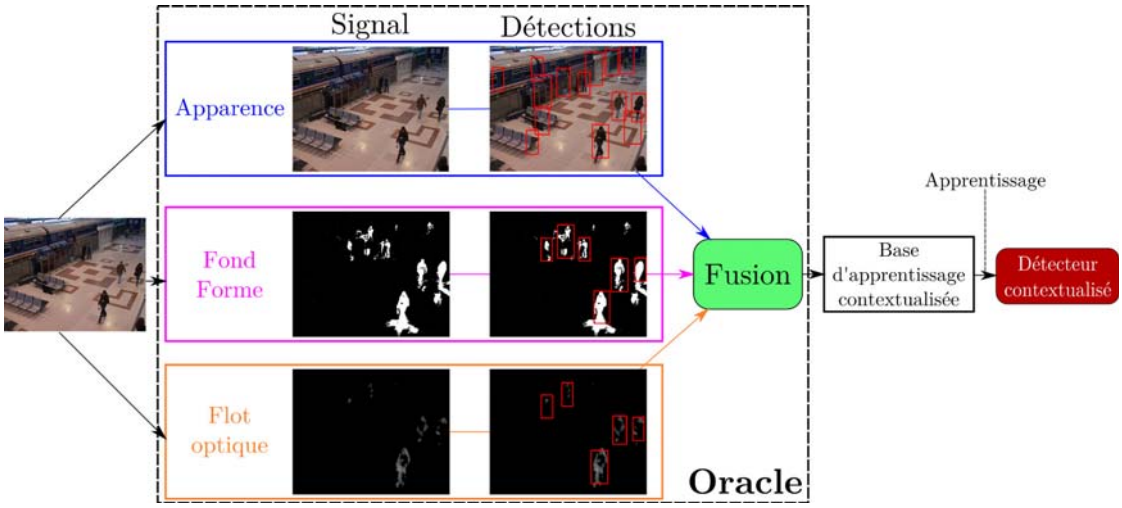


FIGURE 4.1 – **Schéma de fonctionnement de l'oracle.** L'oracle est constitué de trois classifieurs indépendants travaillant sur l'apparence, la segmentation fond/forme et le flot optique. Chacun fournit un ensemble de détections qui doivent être fusionnées (voir la figure 4.2) pour former une base d'apprentissage contextualisée. Le détecteur final est entraîné à partir des données contenues dans cette base.

En résumé, trois classifieurs sont créés puis utilisés indépendamment dans trois détecteurs. Les seules différences dans leur construction se situent au niveau du type d'image et du type de descripteur utilisé pour le classifieur. Tout le reste étant identique. Le premier classifieur basé sur l'apparence utilise un descripteur HOG (*cf* paragraphe 2.3.1.4, page 23). Le second basé sur la soustraction de fond utilise un descripteur s'appuyant sur des ondelettes de Haar classiques (*cf* paragraphe 2.3.1.1, page 20). Enfin le troisième

classifieur utilise le flot optique et un descripteur dérivé de HOF (*cf* paragraphe 2.3.1.4, page 24). Chaque classifieur est entraîné sur une base d'apprentissage générique différente. Lors de la phase d'exploitation de l'oracle, les trois classifieurs fonctionnent de manière indépendante et en parallèle. La partie implémentation de l'oracle (page 77) aborde plus en détails la construction des classifieurs.

À la manière du cotraining (*cf* paragraphe 3.5.1, page 55), les trois classifieurs sont entraînés sur des signaux aussi indépendants que possible. Cela permet, lors d'une étape de fusion, de croiser les réponses de tous ces classifieurs afin de corriger les erreurs de l'un grâce à la sortie des autres.

4.3.2 Fusion des signaux : intersection

Balayer une image à l'aide de ces trois classifieurs permet d'obtenir un score de confiance par signal pour chaque position et échelle testées. Il est ensuite nécessaire de fusionner ces cartes de confiance. Ces classifieurs étant *a priori* indépendants, les scores fournis par chacun d'eux ne sont pas comparables. Deux choix s'offrent alors : travailler directement sur les cartes de confiance après les avoir normalisées pour les rendre comparables, ou bien travailler sur les détections fournies par les classifieurs après les avoir regroupées.

La normalisation des scores d'un classifieur est une possibilité. Par exemple, [Platt, 1999] modifie le score du classifieur pour se rapprocher d'une probabilité d'appartenance à une classe. Nous avons préféré travailler après l'étape de regroupement des boîtes (*cf* paragraphe 2.3.3.4, page 37). Cela amène naturellement à comparer les boîtes finales (et non les scores) obtenues à l'aide des différents classifieurs, ce qui simplifie le problème. Nous allons expliquer comment utiliser pour cela le critère de similarité introduit au paragraphe 2.3.3.4 (page 39) entre deux boîtes après regroupement.

4.3.2.1 Génération des exemples positifs

À la même manière de [Dalal et Triggs, 2005], nous utilisons le mean shift (*cf* paragraphe 2.3.3.4, page 37) pour regrouper toutes les boîtes d'un même classifieur qui ont un score positif. En sortie du regroupement, nous définissons un score pour chaque détection qui est égal à la somme des scores des boîtes qui ont contribué à ce regroupement. Nous obtenons pour une même image un ensemble de détections (boîte + score) par classifieur.

À partir de ces données, il faut collecter des exemples positifs, ce qui implique une étape de fusion des réponses des trois classifieurs génériques. La fusion est délicate car si elle est trop restrictive, des exemples difficiles risquent d'être oubliés. Les exemples difficiles sont des exemples que le classifieur a du mal à labelliser correctement. Ils sont généralement proches de la frontière de décision du classifieur et sont donc intéressants à mettre dans la base car ils peuvent permettre de préciser localement la séparation entre les deux classes. Cependant ajouter ce type d'exemples risque de bruiter la base et d'incorporer dans les exemples positifs de nombreux exemples négatifs qui statistiquement sont plus nombreux.

Pour ne pas détériorer la précision de l'oracle, nous privilégions les détections qui apparaissent en sortie de plusieurs classifieurs et dont le score de confiance est suffisamment élevé. Dans nos expérimentations, nous nous sommes aperçus que le classifieur générique

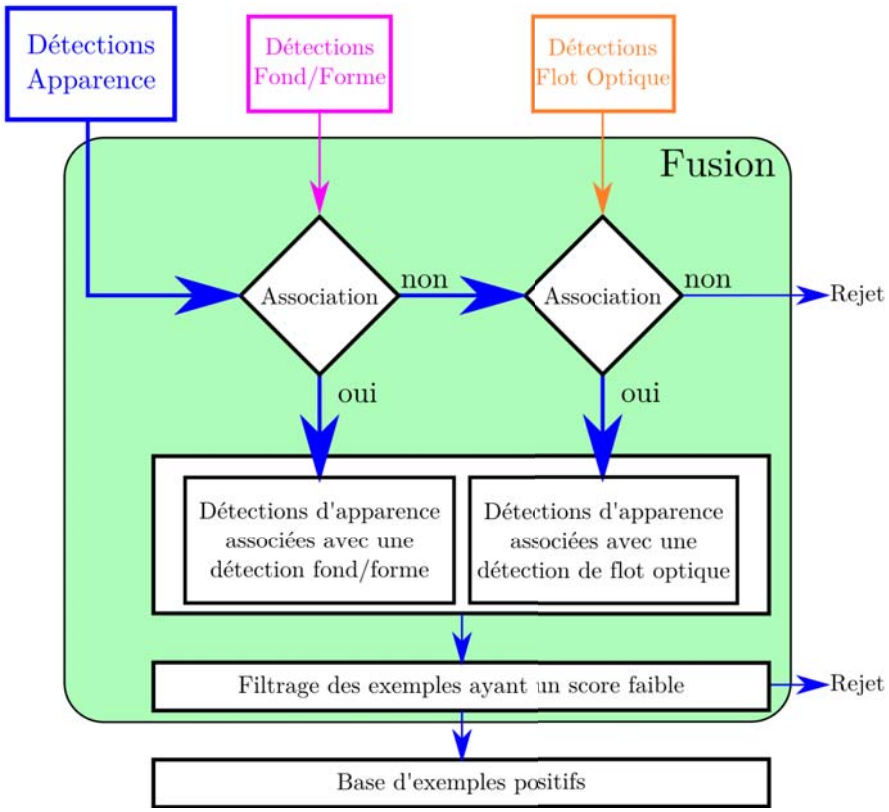


FIGURE 4.2 – Schéma de fonctionnement de la fusion des réponses des classifieurs au sein de l'oracle pour l'obtention de la base d'exemples positifs.

d'apparence était généralement plus précis au niveau de la localisation des détections, tandis que les deux autres classifieurs génériques fournissaient une information plus floue de présence d'un piéton dans une zone assez étendue de l'image. Par exemple une détection de la soustraction de fond peut être décentrée par l'ombre d'un piéton. Nous avons donc décidé de privilégier l'apparence, les autres oracles servant de filtres. Pour cela nous réglons le seuil du classifieur d'apparence sur un seuil assez bas. Il y a donc beaucoup de faux positifs qui sont éliminés grâce à la fusion.

Le choix des exemples positifs est effectué par un vote majoritaire expliqué dans l'algorithme 5 et sur la figure 4.2. Il peut être vu comme la vérification des réponses de l'apparence par la segmentation fond/forme et le flot optique. Une première association est réalisée entre les détections de l'apparence et celles de la segmentation fond/forme. L'association est uniquement basée sur le recouvrement des boîtes comme défini par la métrique choisie. Si la similarité entre deux boîtes est inférieure à 0,5 alors elles ne peuvent pas être appariées. Seules les boîtes d'apparence qui ont pu être appariées sont ajoutées à la base contextualisée. Une deuxième association est ensuite réalisée entre les détections du flot optique et celles non déjà associées de l'apparence. De la même manière les boîtes d'apparence appariées sont incorporées dans la base tandis que les boîtes non associées jusqu'ici sont définitivement écartées.

Algorithme 5: Oracle 2D : Fusion des signaux

Entrées :
Soit App, Sdf et Fo les ensembles de détections des signaux d'apparence, de soustraction de fond et de flot optique.
Soit sim, une mesure de similarité entre détections : $sim(x, y) = \frac{Aire(x \cap y)}{Aire(x \cup y)}$
N le nombre de détections maximales désiré

Sorties :
O l'ensemble des détections de l'oracle

Notations :
Soient X et Y deux ensembles de détections.

$$X \sqcap Y = \{x \in X \mid \exists y \in Y, sim(x, y) > 0,5\}$$
$$X \star Y = \{x \in X \mid \forall y \in Y, sim(x, y) \leq 0,5\}$$

Algorithme :

Initialiser O à $\emptyset$

Intersection avec les détections de la soustraction de fond :
Ajouter $App \sqcap Sdf$ dans O

Intersection avec les détections du flot optique :
Ajouter $(App \star Sdf) \sqcap Fo$ dans O

Garder les N détections de O ayant le meilleur score.

Jusqu'à présent ont été éliminées les détections d'apparence qui correspondent à un objet immobile ou à du fond. Cependant il peut subsister quelques erreurs après cette étape. Pour les filtrer et prendre les exemples les plus significatifs, nous ne considérons, à la manière du self-learning (*cf* paragraphe 3.5.1, page 55), que les N détections ayant le score le plus élevé. Comme toutes les détections proviennent du classifieur appris sur l'apparence, les scores sont issus de cet unique détecteur et sont donc comparables. N représente le nombre d'exemples positifs souhaités au moment de l'apprentissage contextualisé et doit être estimé de façon approximative en tenant compte du nombre de piétons dans la vidéo.

4.3.2.2 Génération des exemples négatifs

Dans le paragraphe précédent, seule la génération des exemples positifs qui sont fournis de manière directe par un oracle a été présentée. Pour construire la base d'apprentissage, il faut générer des exemples négatifs. Notre stratégie consiste à tirer aléatoirement des boîtes dans toutes les images en évitant soigneusement les zones détectées précédemment. L'oracle ayant un rappel faible, tous les piétons ne sont pas détectés. Ils risquent donc de se retrouver dans la base des négatifs. Mais cela est peu probable du fait que les exemples négatifs sont beaucoup plus nombreux que les exemples positifs.

D'autre part, nous avons décidé d'incorporer dans la base des négatifs, des exemples contenant des morceaux de piétons. Le principal intérêt de cet ajout est de rendre le modèle de piéton plus précis et d'améliorer la localisation des détections. Ainsi, le classifieur contextualisé est entraîné pour ne détecter qu'un piéton complet et non une partie. De plus comme la scène traitée est fixe, de nombreuses observations vont être semblables. Cela risque de conduire à une base qui peut ne pas être assez riche avec un manque de variabilité au niveau des exemples négatifs et trop peu d'exemples difficiles. Dans la pratique, cela conduit à prendre des parcelles de l'image qui intersectent des détections fournies par l'oracle. Cependant un exemple positif et un exemple négatif ne doivent pas trop se recouvrir et ne pas vérifier le critère de similarité défini dans le paragraphe 2.3.3.4, donc : $sim(B_{\text{piéton}}, B_{\text{négatif}}) < 0,5$.

La génération des exemples négatifs est commune à tous les oracles.

4.3.3 Implémentation

4.3.3.1 Implémentation de l'oracle

Les caractéristiques de l'oracle 2D sont maintenant étudiées. Il utilise trois signaux distincts. L'apparence est directement issue de l'image couleur. La segmentation fond/-forme et le flot optique sont calculés grâce à la librairie OpenCV. Cette implémentation utilise pour la soustraction de fond un modèle à base de gaussiennes dérivé du travail de [Zivkovic, 2004]. 100 images sont utilisées pour que la soustraction de fond soit correctement initialisée. Pour le flot optique, nous avons décidé d'utiliser l'algorithme de [Black, 1996].

L'oracle est constitué de trois classifieurs génériques, chacun construit à partir de 400 itérations de boosting, RealAdaboost, (*cf* paragraphe 2.3.2.2, page 30) sans cascade. La cascade est principalement utilisée pour accélérer le temps de détection, mais elle complique l'étude du classifieur qui est composé de plusieurs étages indépendants. N'étant pas limité par le temps de calcul et afin de mieux comprendre le fonctionnement du système, nous n'avons pas utilisé de cascade. Comme dans le cas de [Dalal et Triggs, 2005], le modèle appris est monolithique.

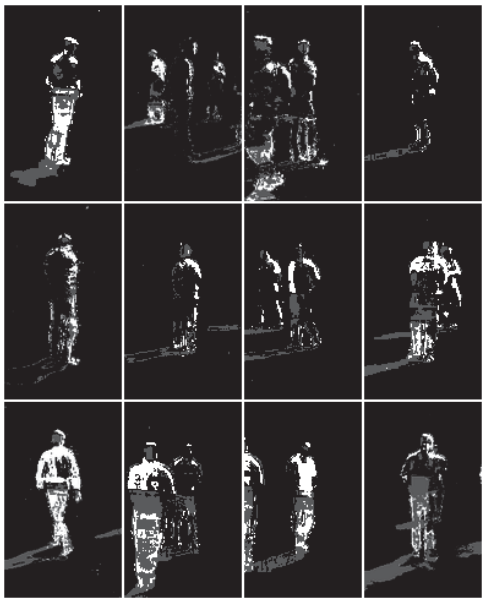
Le classifieur fonctionnant sur l'apparence utilise un descripteur HOG (*cf* paragraphe 2.3.1.4, page 23). Nous avons utilisé le même descripteur pour le classifieur de flot optique : les composantes en x et en y du flot optique correspondent au gradient en x et en y de l'image d'apparence. En revanche, pour la segmentation fond/forme, nous avons décidé d'utiliser des ondelettes de Haar. Celles-ci permettent principalement de coder les contours des objets et non leur densité. Il est en effet impossible avec les ondelettes simples de savoir si une zone homogène correspond à du fond ou un objet. Pour combler cette lacune nous avons ajouté la moyenne de la valeur des pixels du bloc traité.

Nous entraînons le classifieur basé sur l'apparence à l'aide de la base de piétons de l'INRIA (*cf* paragraphe 3.3.2, page 47). Cette dernière ne possédant pas d'information temporelle, les deux autres classifieurs ont été appris sur des bases construites spécialement. Elles contiennent environ 850 exemples positifs et 8 000 exemples négatifs. Les figures 4.3, 4.4 et 4.5 présentent la scène dont ont été extraits les exemples ainsi que des échantillons de ces bases d'apprentissage.

Comme pour tous les classifieurs, le point de fonctionnement de l'oracle est un compromis entre son rappel et sa précision. Nous souhaitons privilégier fortement sa précision mais son rappel ne doit pas être trop faible sinon la base d'apprentissage contextualisée ne contiendrait que des exemples positifs triviaux. Nous avons alors fait le choix de fixer à 0 les seuils de détection de tous les classifieurs génériques qui constituent l'oracle. En pratique, c'est un seuil assez bas, en particulier pour le détecteur d'apparence. Le rappel de ce détecteur d'apparence est donc élevé ce qui permet de ne pas perdre trop de piétons. Au contraire sa précision est assez faible mais la fusion des signaux élimine les erreurs potentielles.

Cet oracle ne travaillant que dans le plan image, la taille et l'orientation des piétons sont inconnues. Pour limiter les temps de calcul lors de nos expérimentations, nous supposons, dans la suite, que les piétons ont une orientation, à peu près constante dans toute l'image. De plus nous considérons que cette orientation est relativement proche de la verticale. Cela implique notamment que le rapport entre la hauteur et la largeur des piétons est d'environ 0,5.

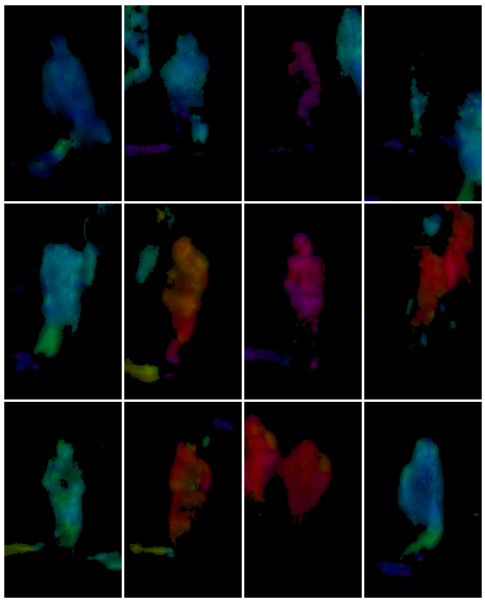
FIGURE 4.3 – Image de la séquence utilisée pour extraire des exemples de soustraction de fond et de flot optique. Comme dans le cas de la base INRIA, les piétons sont vus de face.



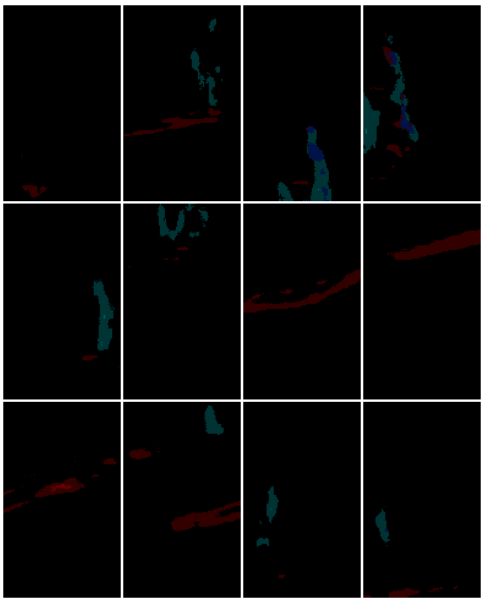
(a) Exemples positifs (soustraction de fond)

(b) Exemples négatifs (soustraction de fond)

FIGURE 4.4 – Échantillon de la base générique de soustraction de fond.



(a) Exemples positifs (flot optique)



(b) Exemples négatifs (flot optique)

FIGURE 4.5 – Échantillon de la base générique de flot optique.

4.3.3.2 Implémentation du classifieur contextualisé

Une fois la base d'apprentissage contextualisée générée, il faut construire le classifieur contextualisé. Afin d'étudier, toutes choses égales par ailleurs, l'effet de la contextualisation sur l'apprentissage, nous avons décidé d'employer la même approche pour le classifieur d'apparence générique de l'oracle et le classifieur contextualisé du détecteur final. Ce dernier n'utilise que l'apparence et il est entraîné de la même façon que le classifieur de l'oracle basé sur ce signal. Seule la base d'apprentissage diffère entre les deux, puisque seuls des exemples de la scène sont utilisés pour fabriquer le classifieur final. Nous utilisons donc toujours l'algorithme RealAdaboost. Pour limiter le sur-apprentissage, le nombre de rounds de boosting a été limité, chaque modèle est monolithique et fabriqué à l'aide de 400 classifieurs faibles. 1 800 exemples positifs et 8 000 négatifs sont conservés lors du filtrage après la fusion des classifieurs et sont utilisés pour l'apprentissage.

Il n'est pas obligatoire d'entraîner le classifieur d'apparence générique et le classifieur contextualisé de la même manière. D'autres approches ont été proposées dans la littérature pour entraîner le classifieur contextualisé. Lors de l'apprentissage, le boosting tend à augmenter le poids des exemples mal classés afin d'obliger les classifieurs faibles à les classer correctement. Puisque le poids de ces exemples est très supérieur à celui des autres, l'apprentissage va logiquement les privilégier. Dans le cas d'Adaboost, ce phénomène est encore accentué par la fonction de coût exponentielle. Malheureusement, dans la pratique, il est impossible d'obtenir automatiquement une base sans erreur de label. Les exemples mal labellisés, donc très difficiles à classer correctement, vont mécaniquement obtenir un poids élevé et risquent de perturber l'apprentissage.

Dans son travail [Wu et Nevatia, 2007b], limite le poids possible des exemples pour éviter cette situation. Une autre possibilité est d'utiliser un algorithme plus robuste au bruit comme ceux basés sur le *Multiple Instance Learning* comme MILBoost [Viola *et al.*, 2006]. Ces approches ne considèrent plus les exemples séparément, mais les regroupent en paquets. Si un paquet contient un exemple positif alors tout le paquet est considéré comme positif. Dans le cas contraire il est labellisé comme négatif.

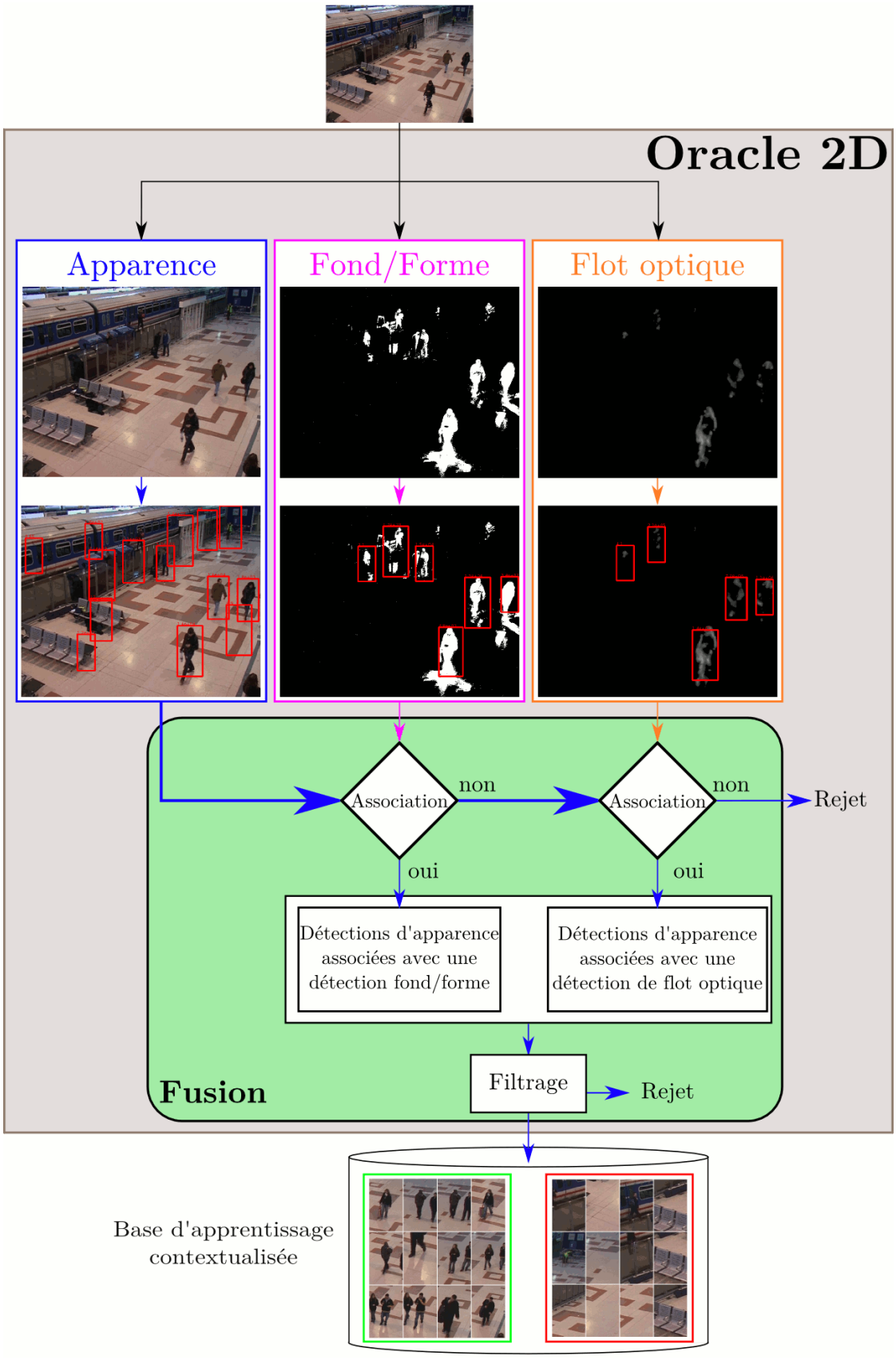


FIGURE 4.6 – Schéma complet de l'oracle 2D. Il présente l'architecture globale de l'oracle 2D et la procédure de construction des exemples positifs de la base contextualisée.

4.3.4 Validations expérimentales 2D

Maintenant que l'oracle a été défini, il est nécessaire de valider expérimentalement notre approche de contextualisation du détecteur sur plusieurs vidéos.

4.3.4.1 PETS 2006

Ici nous travaillons sur la vue 4 du corpus PETS 2006 pour fabriquer un classifieur. Le détecteur final est entraîné avec des exemples provenant d'une séquence (S2-T3-C). Les classifieurs, y compris ceux constituant l'oracle, sont testés sur environ 1 000 images provenant d'une autre séquence partageant le même point de vue (S7-T6-B).

(a) Exemples positifs (b) Exemples négatifs

FIGURE 4.7 – Exemples choisis provenant de la base d'apprentissage du détecteur final pour PETS 2006

La figure 4.7 montre un échantillon de la base d'apprentissage obtenue après fusion des classifieurs constituant l'oracle. Pour les exemples positifs, la très grande majorité des imagettes correspond effectivement à un piéton. Cependant il existe deux problèmes principaux :

- Lorsque plusieurs piétons sont proches, le regroupement ne parvient pas toujours à les séparer correctement et a tendance à mal aligner l'exemple,
- La taille de la boîte de l'exemple n'est pas toujours adaptée à l'objet réel.

La taille des imagettes peut être trop grande ou trop petite par rapport au piéton qu'elles contiennent. Cela s'explique par le fait que le problème de la détection est mal contraint en 2D et qu'il n'y a aucune façon pour notre détecteur de connaître même

approximativement la taille d'un objet pour une position donnée dans la scène. D'autre part l'algorithme du mean shift a un grand pouvoir de regroupement puisqu'il arrive à moyenner toutes les boîtes qui ont été détectées pour un même piéton. Or généralement il y a plus de petites boîtes que de grandes boîtes. Le regroupement conduit donc ici à rétrécir les boîtes sur lesquelles le classifieur contextualisé est appris. En particulier les marges autour des piétons ne sont plus respectées (*cf* paragraphe 2.3.1, page 20).

Comme souhaité, les exemples négatifs correspondent pour la plupart soit à des observations sans piétons, soit à des observations avec des morceaux de piétons.

Sur cette séquence d'apprentissage (S2-T3-C), l'oracle obtient un rappel de 0,16 avec une précision de 0,99. Ces valeurs ont été calculées après le filtrage sur les scores des détections d'apparence. Comme espéré, il possède une précision très élevée. Celle-ci est obtenue sans connaissance *a priori* de la scène et sans seuil à régler puisque tous les classifieurs de l'oracle ont un seuil de détection par défaut fixé à 0.

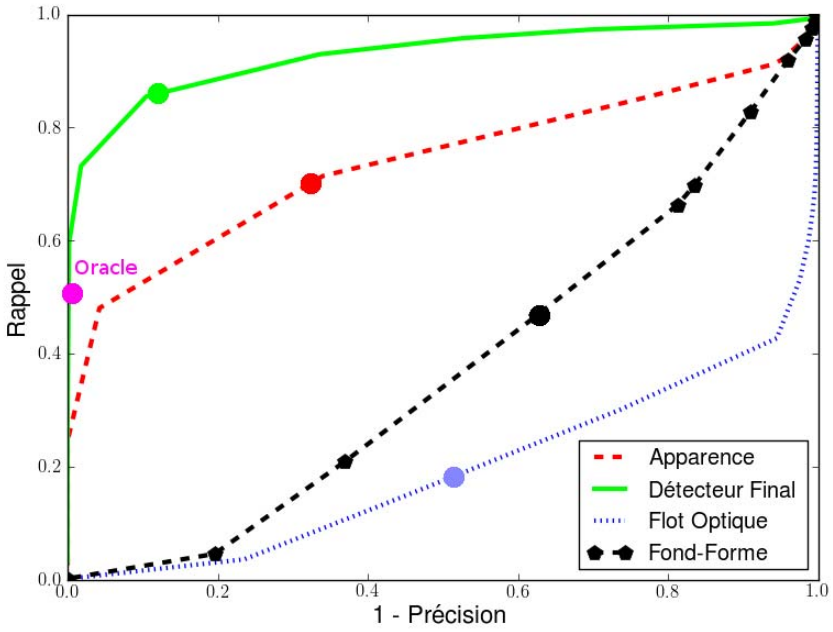


FIGURE 4.8 – Courbes précision-rappel sur la séquence PETS 2006 pour les 3 classifieurs constituant l'oracle et pour le détecteur final. Les cercles indiquent les seuils pour lesquels la F-Mesure est maximale pour le classifieur.

La figure 4.8 montre les courbes précision-rappel pour chaque classifieur constituant l'oracle ainsi que celle du détecteur final. Comme annoncé ce dernier est plus performant que le classifieur générique d'apparence. La contextualisation a bien permis d'améliorer significativement les performances. Comme les classifieurs de soustraction de fond et de flot optique ne sont pas très précis en terme de position, leurs performances sont en deçà des autres classifieurs. Ils permettent néanmoins de supprimer la majorité des faux positifs immobiles du classifieur générique d'apparence.

La table 4.2 indique la précision et le rappel de chaque classifieur pour le point où

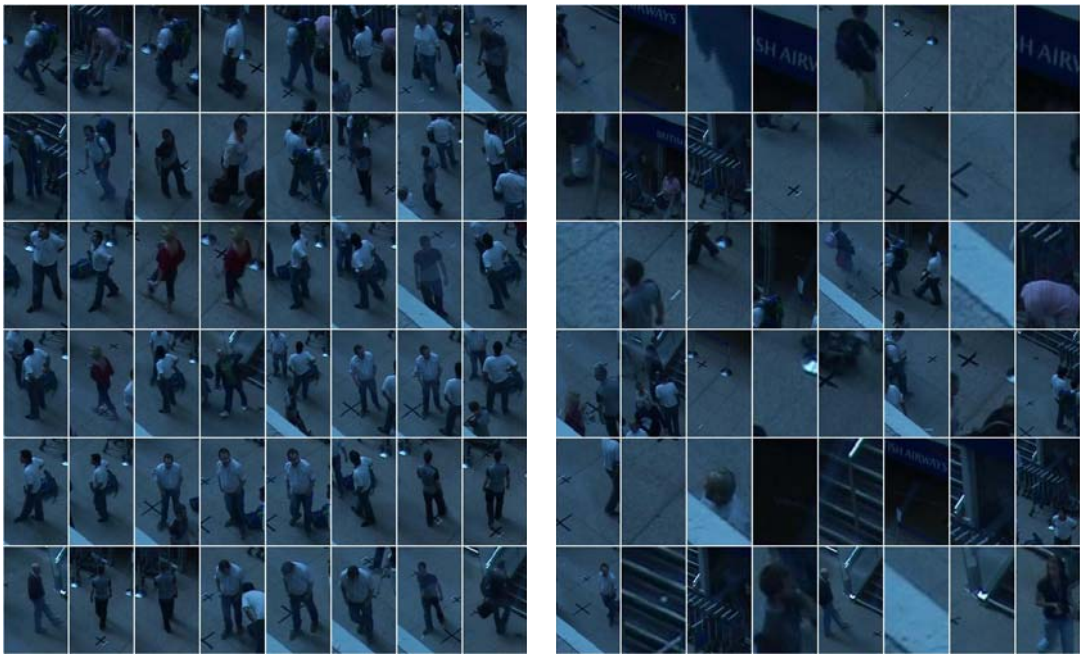
la F-Mesure est maximale. Remarquons que la précision des classifieurs appris sur la segmentation fond/forme et sur le flot optique est assez faible. Comme expliqué lors de l'étape de fusion, cela peut s'expliquer par le fait que ces classifieurs sont moins précis en position, c'est-à-dire que leurs détections sont plus dispersées autour de la cible. Souvent deux cibles proches sont regroupées en une seule et la position de chacune est mal estimée. Ils fournissent donc une information de présence d'un piéton dans une zone de l'image, tandis que le classifieur basé sur l'apparence apporte une information de localisation plus précise.

TABLE 4.2 – Caractéristiques des détecteurs sur la séquence PETS 2006 (S7-T6-B-4), au point de F-Mesure maximale.

	Rappel	Précision	F-Mesure
Apparence	0,71	0,66	0,69
Fond/Forme	0,47	0,38	0,42
Flot Optique	0,30	0,26	0,28
Oracle	0,49	**0,99**	0,65
Classifieur final contextualisé	**0,85**	0,90	**0,87**

4.3.4.2 PETS 2007

Le second corpus sur lequel nous avons testé notre méthode est la vue 3 de PETS 2007.



(a) Exemples positifs (b) Exemples négatifs

FIGURE 4.9 – Exemples choisis provenant de la base d'apprentissage du détecteur final pour PETS 2007

Contrairement à PETS 2006, le point de vue de la caméra est ici très différent de celui qui a servi à créer la base d'apprentissage générique. Pour la base d'apprentissage, les piétons sont pris de face et sont bien verticaux, alors que dans le cas de PETS 2007 ils sont observés de haut et sont généralement inclinés. La base spécialisée a été construite sur la séquence 3, tous les détecteurs sont évalués sur les 1 000 premières images de la séquence 5. La figure 4.9 montre un échantillon de la base d'apprentissage obtenue après fusion des classifieurs constituant l'oracle. La figure 4.10 montre les courbes précision-rappel pour chaque classifieur constituant l'oracle ainsi que celle du détecteur final. Les points figurant sur ces courbes indiquent le point de fonctionnement optimal au sens de la F-Mesure pour les classifieurs.

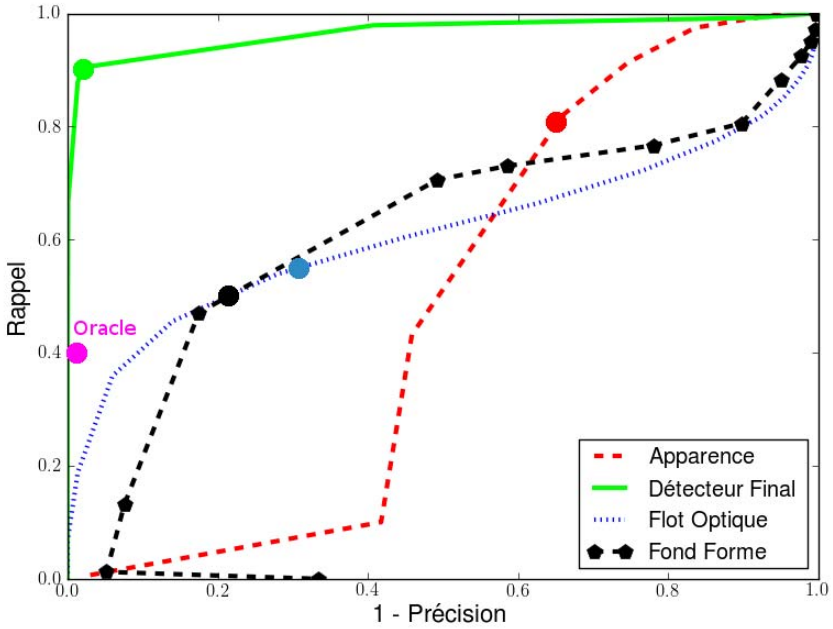


FIGURE 4.10 – Courbes précision-rappel sur la séquence PETS 2007 pour les 3 classifieurs constituant l'oracle et pour le détecteur final. Les cercles indiquent les seuils pour lesquels la F-Mesure est maximale pour le classifieur.

TABLE 4.3 – Caractéristiques des détecteurs sur la séquence PETS 2007 (S05 - 3)

	Rappel	Précision	F-Mesure
Apparence	0,82	0,34	0,48
Fond/Forme	0,47	0,82	0,60
Flot Optique	0,54	0,72	0,62
Oracle	0,40	**0,99**	0,57
Classifieur final contextualisé	**0,90**	0,98	**0.94**

De la même manière que pour PETS 2006, la table 4.3 contient la précision et le rappel de chaque classifieur pour le point où la F-Mesure est maximale.

Contrairement au cas précédent, les classifieurs basés sur la segmentation fond/forme

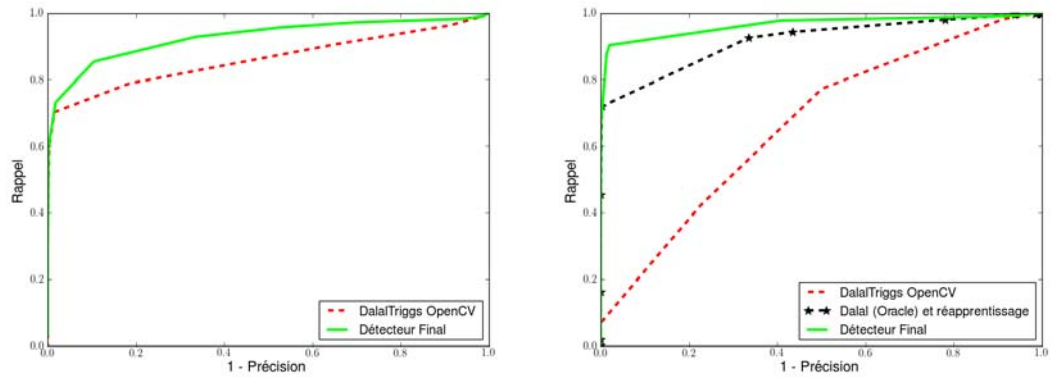
et sur le flot optique sont meilleurs que celui appris sur l'apparence. Cela s'explique par les deux faits suivants :

- Les exemples provenant de cette séquence ont des apparences trop différentes de ceux issus de la base générique. Un classifieur basé uniquement sur ce signal offre donc de mauvaises performances.
- Il y a des groupes de personnes dans cette séquence. Les classifieurs qui ne sont pas discriminants (segmentation fond/forme et flot optique) peuvent peut-être associer une détection éloignée du piéton qui l'a générée, à celui d'à côté. Bien que cela soit une erreur en pratique, notre méthode d'évaluation ne les pénalise pas.

Comme les courbes le montrent, sur ces deux séquences, le détecteur final contextualisé permet d'obtenir de bien meilleures performances, à la fois en rappel et en précision, que le classifieur de l'oracle basé sur l'apparence.

De plus, il ne semble pas y avoir de lien entre les résultats du classifieur générique d'apparence et celui du classifieur final contextualisé. Sur PETS 2007, le classifieur générique offre des performances très médiocres et la contextualisation permet une amélioration importante. Sur PETS 2006, le classifieur générique fonctionne assez bien dès le départ et l'apport de la contextualisation est en conséquence un peu moins important. Il reste néanmoins bien réel et intéressant.

4.3.4.3 Performances du détecteur contextualisé



(a) PETS 2006 (b) PETS 2007

FIGURE 4.11 – Courbes précision-rappel sur les séquences PETS 2006 et 2007 pour notre détecteur final(vert), celui de Dalal et Triggs(rouge) et celui utilisant l'oracle basé sur le classifieur de Dalal (noir)

Dans cette section nous comparons les performances du détecteur obtenu avec notre oracle et celles d'un détecteur de l'état de l'art. Nous avons retenu l'implémentation du détecteur de Dalal et Triggs [Dalal et Triggs, 2005] présent dans OpenCV. Nous appliquons la même méthode d'évaluation que précédemment. Pour prouver l'utilité de l'oracle, nous avons aussi entraîné un classifieur à l'aide d'une base contextualisée extraite grâce au détecteur de Dalal et Triggs. Ce dernier fournit la position des piétons tandis

que les exemples négatifs sont obtenus aléatoirement. L'apprentissage est le même dans tous les cas.

Les courbes de la figure 4.11 présentent les résultats des expériences sur les deux séquences retenues dans cette étude. Elles prouvent que lorsque la base d'apprentissage et la scène traitée sont trop différentes (PETS 2007), un détecteur contextualisé permet d'améliorer significativement les résultats. Dans le cas d'une scène dont les piétons ressemblent à ceux de la base (PETS 2006), les gains sont plus minimes mais tout de même présents.

4.3.5 Améliorations possibles

Nous venons de présenter un premier oracle 2D dont le rôle est de détecter des piétons en faisant le minimum d'erreur. Deux de ses faiblesses ont aussi été évoquées (rappel faible, seuil de détection fixe et relativement faible du classifieur d'apparence).

Les paragraphes suivants explorent des modifications possibles de notre oracle en vue de l'améliorer. La première vise à augmenter le rappel de l'oracle et consiste à ajouter un module de tracking en sortie de l'oracle. L'idée est d'ajouter des exemples qui n'ont pas été détectés par l'oracle et qui sont donc potentiellement difficiles à traiter. La deuxième amélioration porte sur l'adaptation du seuil de détection du classifieur d'apparence générique. Le but est de réduire fortement les faux positifs qu'il serait ensuite nécessaire de filtrer en sortie de ce classifieur et ainsi d'augmenter la précision de l'oracle.

Toutes les expériences concernées sont réalisées sur la séquence ViCoMo 1, car elle présente plusieurs difficultés. Elle contient de nombreux piétons qui peuvent donc s'occulter, ainsi que des structures verticales dont certaines ressemblent à un humain comme les mannequins en bas à gauche.

4.3.5.1 Oracle et tracking

La première amélioration potentielle évaluée est la combinaison de l'oracle et du tracking. L'oracle ne prend que très peu en compte l'aspect temporel de la vidéo. Seuls la soustraction de fond et le flot optique, utilisés au travers de leurs classifieurs respectifs reposent dessus. Le déplacement d'un piéton dans l'image est un processus s'étalant sur de nombreuses images consécutives. L'aspect temporel du signal vidéo est donc très important et il pourrait servir à augmenter le rappel de l'oracle. Il serait ainsi possible d'acquérir des exemples que l'oracle n'a pas retenus. Il peut s'agir de piétons immobiles non repérés par la soustraction de fond ou le flot optique. Mais les plus intéressants sont sans doute les piétons non repérés par le détecteur d'apparence qui sont donc difficiles à classifier. La figure 4.12 présente l'architecture de cet oracle.

Un module de tracking multi-cibles exploite justement le signal temporel d'une vidéo afin d'y suivre un objet. Dans notre cas, il permet de faire la jonction entre les différentes détections de l'oracle. Comme nous disposons déjà de détections et que nous voulons compléter les pistes entre deux détections consécutives, il parait intéressant de s'intéresser au tracking par détections. L'état de l'art regorge de méthodes de tracking multi-cibles basées sur des détections. Notre choix s'est porté sur l'approche markovienne de [Breitenstein *et al.*, 2011] qui a été évaluée par son auteur sur de nombreuses séquences.

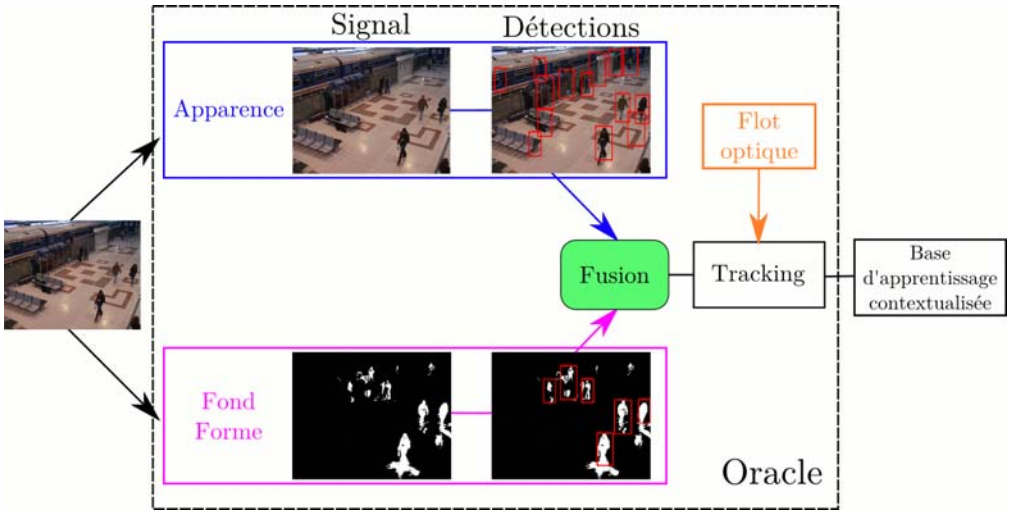


FIGURE 4.12 – Schéma de fonctionnement de l'oracle avec tracking

Un tracker est généralement composé d'un modèle d'apparence et d'un filtre. Le premier a pour rôle de représenter les caractéristiques de l'objet suivi. Il repose donc bien souvent sur un descripteur couleur, parfois associé à un algorithme d'apprentissage qui permet de mettre en lumière les composantes qui discriminent le mieux l'objet par rapport au reste de l'image ou aux autres objets suivis. Dans le cas de [Breitenstein *et al.*, 2011], le modèle d'apparence est principalement constitué par un descripteur d'histogrammes couleurs dans l'espace, couplé avec un algorithme d'apprentissage supervisé de boosting online (*cf* partie 6.3.2.1, page 147 pour plus de détails sur le boosting online). Afin d'apprendre ce modèle d'apparence pour une cible il est donc nécessaire de construire une base d'apprentissage dédiée à la cible. Les exemples positifs sont constitués de l'objet suivi par ce tracker, tandis que les exemples négatifs sont fabriqués à l'aide du voisinage de cet objet mais aussi directement des autres objets suivis dans la scène. Ainsi le modèle d'apparence ne doit répondre positivement que sur l'objet d'intérêt, même lorsque celui-ci croise un autre piéton qui lui ressemble.

Le second élément qui compose le tracker est le filtre. Ce dernier a la charge de retrouver la position de l'objet suivi dans l'image courante connaissant son état aux instants passés. [Breitenstein *et al.*, 2011] utilise un filtre particulaire de type *Sampling Importance Resampling* (ou SIR), qui a été proposé initialement par [Rubin *et al.*, 1988] et popularisé pour le suivi par [Isard et Blake, 1998].

Un tracker par détections multi-cibles fonctionne de pair avec un détecteur de piétons. Le tracker est initialisé grâce aux positions fournies par le détecteur. Dans les images suivantes, les nouvelles détections servent à contraindre le tracker sur les positions des piétons détectés et à vérifier la validité du suivi au cours du temps. Une association entre les boîtes fournies par les trackers et celles du détecteur est donc réalisée à chaque image. Elle prend en compte des critères géométriques (position et taille des boîtes) et des critères d'apparence : le modèle du tracker doit être compatible avec le contenu de la détection. Si le tracker est associé à une détection, la confiance dans le suivi est considérée comme suffisamment importante pour que son modèle d'apparence soit mis à jour avec la nouvelle position. Le traitement continue alors avec l'image suivante. En cas de non

association, le filtre particulaire recherche la position optimale de l'objet dans la nouvelle image. Cependant, dans ce dernier cas, le modèle d'apparence n'est pas mis à jour avec les nouvelles données. Plus le temps entre deux associations est grand, plus la probabilité que le tracker perde sa cible est grand et il est donc détruit au bout de quelques images.

Le code de [Breitenstein *et al.*, 2011] n'étant pas disponible, nous avons dû nous inspirer de sa méthode pour créer notre propre implémentation et l'adapter à nos besoins. Le détecteur de piétons utilisé par le tracker est remplacé par l'oracle qui fournit des détections basées sur le signal d'apparence qui ont été filtrées par celles basées sur la soustraction de fond. Un tracker est lancé lorsque deux détections issues de l'oracle sont proches spatialement et temporellement. Pour conserver la précision élevée du système et ne pas initialiser un tracker avec de mauvaises données, nous avons ajouté une phase supplémentaire de validation des détections d'apparence directement par le flot optique. Si un certain pourcentage de pixels contenus dans la boîte d'apparence se sont déplacés, alors un nouveau tracker est créé. Une piste est automatiquement terminée si elle sort de la scène ou si elle n'est pas associée pendant 20 images consécutives. La figure 4.13 présente quelques images de notre tracker en fonctionnement.

FIGURE 4.13 – Résultats issus de notre tracker sur ViCoMo 1. Les trackers sont indiqués par des boîtes et des trajectoires en couleurs. Un tracker associé à une détection est représenté par une boîte entièrement colorée, sinon seul le contour de la boîte apparaît. Les détections non associées sont représentées par des traits noirs et plus fins.

Néanmoins, l'utilisation du tracking ne peut apporter quelque chose que si celui-ci commet peu d'erreurs et si le rappel de l'oracle est faible. En effet, il est contre-productif de filtrer au maximum les sorties de l'oracle pour obtenir une précision élevée si dans un deuxième temps, le tracking en essayant d'élargir la base augmente le taux d'erreur. Il est donc nécessaire que le module de tracking ajoute uniquement des exemples dont il est sûr.

Ainsi la base contient uniquement des exemples issus du tracking, soit associés à une détection de l'oracle, soit situés temporellement entre deux associations consécutives. Grâce à ce filtrage, les fins de pistes (le piéton sort de l'image, le tracker a perdu la cible...) qui, par définition ont une grande incertitude, ne sont pas incorporées dans la base. De même, les détections isolées qui n'ont pas conduit à la création d'un nouveau tracker sont rejetées. Enfin notons que nous n'avons pas besoin d'un tracker parfait, c'est-à-dire d'un tracker qui arrive à suivre tous les piétons pendant longtemps. Un tracker qui suit uniquement un objet pendant quelques images peut être suffisant s'il a été associé convenablement avec des détections pendant cette période. Un autre tracker prendra la relève pour continuer le suivi dans de bonnes conditions.

La suite de ce paragraphe est consacrée à l'évaluation de ce nouvel oracle. Nous avons utilisé le même protocole que précédemment sur cette séquence. Les 10 000 premières images servent à la création de la base. Les images numérotées de 15 000 à 16 000 sont utilisées pour la validation. Afin d'évaluer les améliorations, il faut d'abord connaître la performance du classifieur contextualisé obtenu grâce à l'oracle classique sur cette séquence.

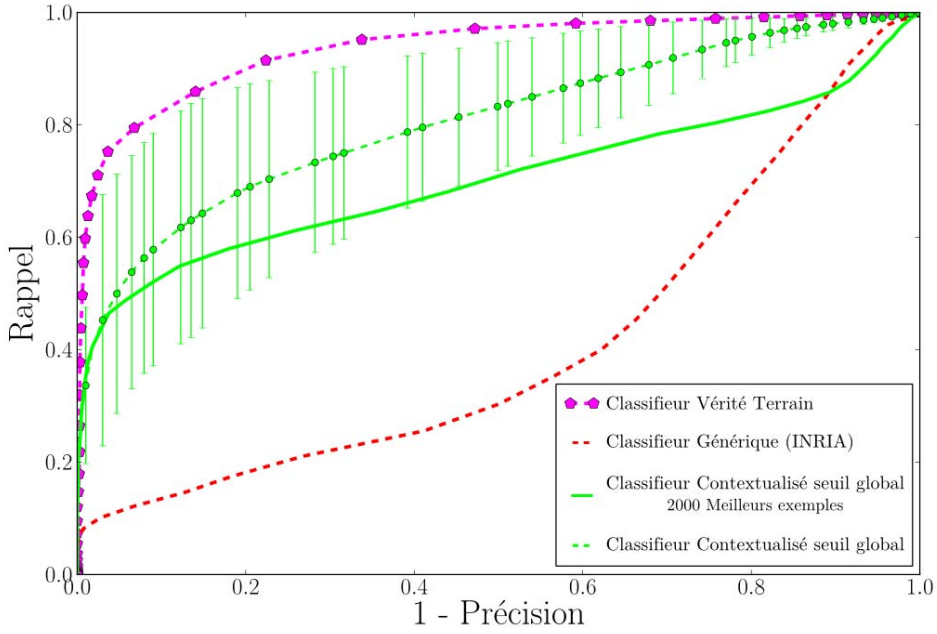


FIGURE 4.14 – Courbes précision-rappel sur la séquence ViCoMo 1 pour les classifieurs générique ou contextualisé. Les barres d'erreur représentent l'intervalle de rayon 2σ.

Les courbes de la figure 4.14 présentent les performances de différents classifieurs 2D. En rouge figure le classifieur générique. Ses performances sont très mauvaises sur cette séquence. En magenta est représenté le classifieur appris à l'aide de la vérité terrain. Cette courbe fournit une indication sur les performances atteignables. Il ne s'agit pas forcément d'une borne supérieure car de nombreux paramètres autres que les exemples d'apprentissage peuvent influencer la contextualisation. Les deux courbes vertes représentent les performances de deux classifieurs contextualisés avec un oracle. Celui en trait plein a été entraîné avec les 2 000 meilleurs exemples positifs. De la même façon que pour PETS 2006 et PETS 2007, la contextualisation par l'oracle apporte un gain significatif sur les performances par rapport au classifieur générique.

La courbe en pointillés correspond à la moyenne de cinq classifieurs contextualisés dont les exemples positifs ont été sélectionnés aléatoirement parmi ceux fournis par l'oracle. Les barres d'erreur représentent l'intervalle de rayon 2σ ayant pour centre un point de la courbe. Sur cette séquence choisir les meilleurs exemples positifs n'apporte rien. Une explication possible est que certains faux positifs ont un score de classification élevé et peuvent donc entrer dans la base. D'autre part certains piétons restent longtemps dans l'image et si leur score de classification est élevé, ils peuvent se retrouver sur-représentés dans la base et perturber l'apprentissage.

Les performances des classifieurs obtenus avec l'oracle classique sont désormais connues sur cette séquence. Celles qui peuvent être atteintes grâce à l'ajout du tracking sont indiquées sur la figure 4.15. Les courbes rouge, magenta et verte (sélection aléatoire) sont les mêmes que celles présentées précédemment. Les résultats du classifieur contextualisé grâce à l'oracle couplé au tracking sont en turquoise. La courbe représente

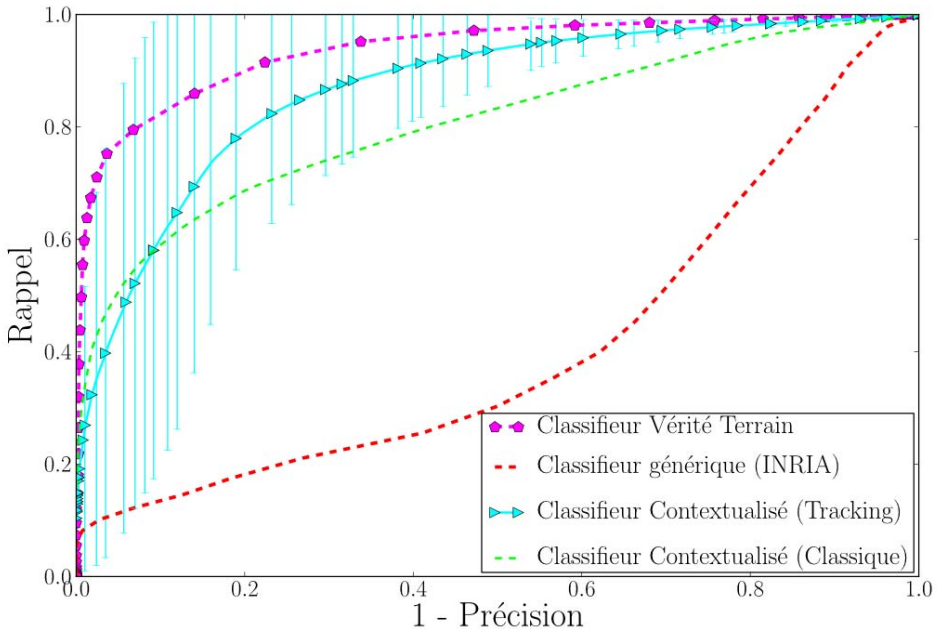


FIGURE 4.15 – Courbes précision-rappel sur la séquence ViCoMo 1. Le classifieur contextualisé grâce à l'oracle couplé au tracking est représenté en turquoise.

la moyenne des performances de cinq classifieurs entraînés avec une sélection aléatoire d'exemples provenant de la même base fournit par l'oracle.

En fin de partie, le tableau 4.4 (page 94) récapitule les différents points de fonctionnement précision et rappel des oracles, ainsi que les points optimaux des classifieurs contextualisés. Dans le cas où l'expérience a été réalisée plusieurs fois, le point de fonctionnement est en fait le point optimal de la courbe moyennée. Comme attendu, le rappel de l'oracle couplé au tracking est meilleur que celui de l'oracle classique. Plus surprenant, sa précision augmente aussi. Ceci est dû au fait que les détections isolées qui ne parviennent pas à lancer un tracker sont ignorées ce qui contribue à éliminer certaines erreurs qui n'ont pas été filtrées par les classifieurs de soustraction de fond et de flot optique.

Au final, l'ajout d'un tracking semble apporter une amélioration nette des performances. Le point de fonctionnement des classifieurs contextualisés à l'aide des oracles classique ou lié au tracking se situent tous les deux autour de 80% de précision mais le classifieur qui découle de l'oracle couplé avec le tracking a un meilleur rappel. Néanmoins cela semble se faire au détriment de la régularité des résultats. La variance du classifieur contextualisé à l'aide du tracking est très importante, en particulier pour les précisions élevées, donc celles qui sont intéressantes en pratique. Face à la complexité de la mise en œuvre du tracking et la variance des résultats obtenus, cette approche bien qu'améliorant les résultats en moyenne, semble trop fragile à l'heure actuelle.

4.3.5.2 Oracle et seuil adaptatif

Ce paragraphe présente la deuxième amélioration proposée : l'intégration d'un seuil de détection adaptatif dans le détecteur générique d'apparence. En effet, l'un des problèmes soulevés lors de la constitution de l'oracle se trouve être le seuil fixé par défaut à 0. Il reste constant pour toutes les séquences testées mais se révèle généralement trop bas. L'avantage de ce seuil bas est qu'il permet de détecter le maximum de piétons qui peuvent ensuite être validés par les classifieurs de soustraction de fond et de flot optique. Son inconvénient est qu'il augmente le nombre de fausses détections, celles-ci n'étant pas toujours aisées à filtrer par la suite. De plus le classifieur générique n'étant pas entraîné avec des données de la scène, le seuil de détection optimal peut varier très fortement d'une partie de l'image à l'autre. Pour pallier ce problème, l'idée serait de calculer des seuils adaptatifs en fonction de la scène et si possible de l'endroit considéré dans l'image. La figure 4.16 présente l'architecture de cet oracle.

Inspirée par les *classifier grids* (*cf* paragraphe 2.3.3.3, page 36), l'estimation de seuils locaux consiste à calculer un seuil pour chaque position dans l'image, permettant ainsi de disposer d'un réglage optimal du classifieur en tout point. La méthode proposée par [Stalder *et al.*, 2010] repose sur le principe suivant. Par définition, le fond est statique et les scores de classification doivent être globalement stables dans le temps. Si à un endroit, la sortie du classifieur est quasiment constante, alors il est peu probable qu'il s'agisse d'un piéton, même si cette sortie est élevée. Par contre lorsqu'un piéton apparaît, le classifieur est sensé répondre fortement, conduisant normalement à une augmentation brusque et locale du score de classification qui peut cependant rester bien inférieur au seuil de détection fixé.

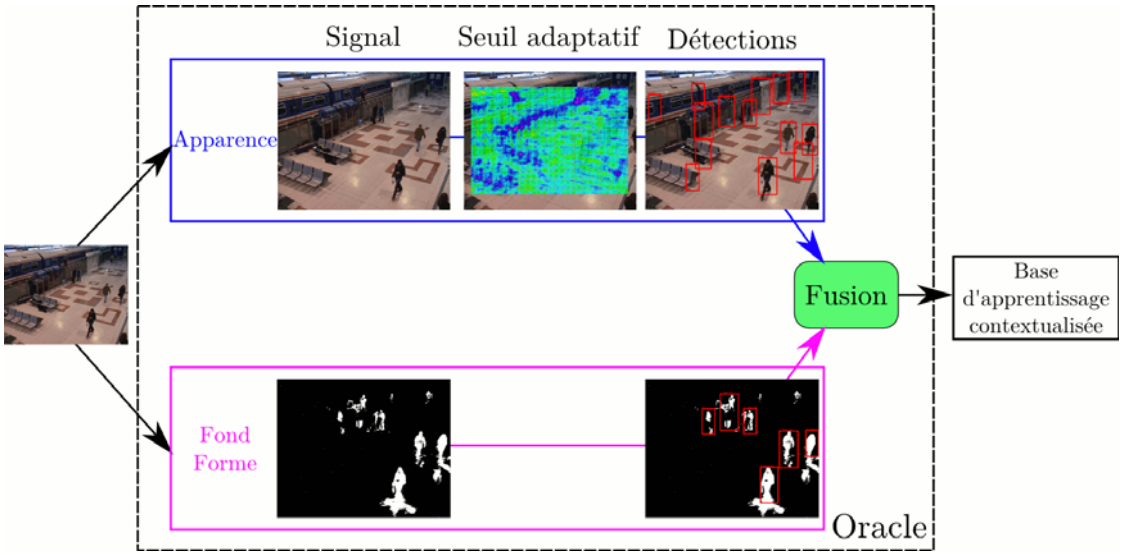


FIGURE 4.16 – Schéma de fonctionnement de l'oracle avec seuil adaptatif

[Stalder *et al.*, 2010] propose donc d'exploiter les variations temporelles des scores de classification, en appliquant un algorithme de soustraction de fond de type [Stauffer et Grimson, 1999], non pas directement sur l'image en couleur, mais sur la carte de scores du classifieur. Grâce à cette astuce, il est possible de capter les variations locales des sorties du classifieur et d'y apporter une réponse adéquate. Coupler cette approche avec l'oracle permet de le contextualiser partiellement. Ainsi il est mieux adapté à la scène et donc plus filtrant. Néanmoins, les seuils adaptatifs ne sont pas parfaits et possèdent un peu les mêmes travers que la soustraction de fond, à savoir une tendance à surdétecter les objets mobiles. Ces objets seront alors très difficiles à éliminer avec les classifieurs de soustraction de fond et de flot optique.

L'implémentation de la méthode est quasiment directe puisqu'il suffit d'employer l'algorithme de soustraction de fond de [Stauffer et Grimson, 1999] sur les cartes de scores du classifieur (une carte par échelle testée). Les paramètres sont les mêmes que ceux de l'algorithme classique.

La figure 4.17 présente en vert foncé les performances du classifieur contextualisé avec cette approche. Une fois encore, l'oracle a été lancé une fois pour construire une base d'apprentissage qui a servi à entraîner cinq classifieurs différents dont les résultats (moyennes et variances) sont indiqués comme précédemment. Afin de comparer les résultats, les courbes précédentes ont aussi été portées sur le schéma.

La table 4.4 résume les performances des différents systèmes présentés dans cette partie en indiquant leurs points de fonctionnement optimaux.

Tout d'abord, il est facile de constater que les classifieurs contextualisés grâce à l'oracle couplé avec les seuils adaptatifs sont les meilleurs et se rapprochent du classifieur contextualisé à la main. Ils possèdent la F-Mesure la plus élevée en moyenne, ainsi que la variance la plus faible dans les résultats autour de leur point de fonctionnement optimal.

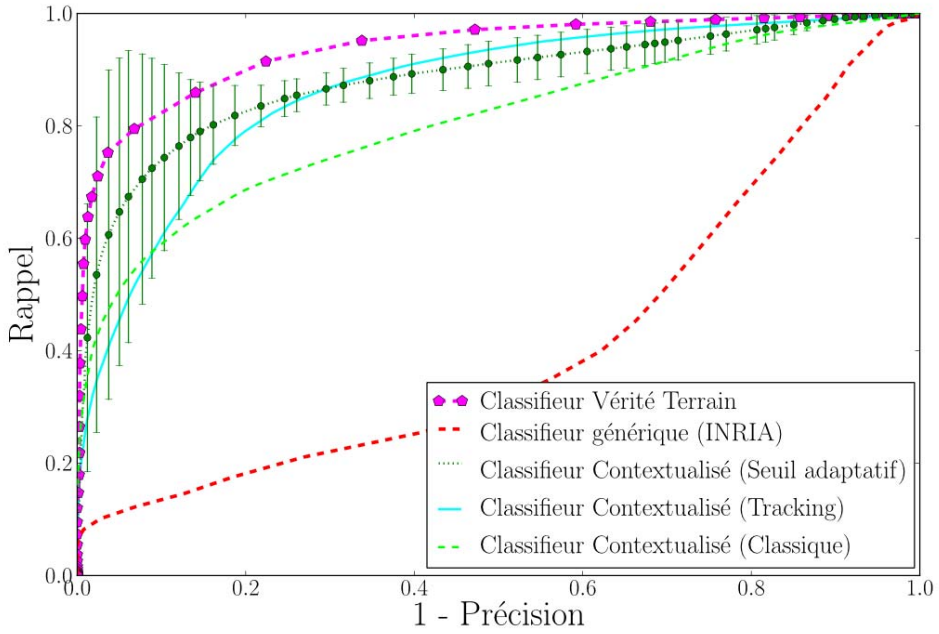


FIGURE 4.17 – Courbes précision-rappel sur la séquence ViCoMo 1. Le classifieur contextualisé grâce à l'oracle couplé au seuil adaptatif est représenté en vert foncé.

TABLE 4.4 – Caractéristiques des oracles et des détecteurs contextualisés sur la séquence ViCoMo 1

	Rappel	Précision	F-Mesure
Oracle classique	0,54	0,77	0,63
Oracle + tracking	**0,60**	0,85	**0,70**
Oracle + seuil adaptatif	0,37	**0,92**	0,53
Classifieur Contextualisé (classique)	0,68	0,80	0,74
Classifieur Contextualisé (tracking)	**0,79**	0,80	0,80
Classifieur Contextualisé (seuil adaptatif)	**0,79**	**0,86**	**0.82**

Cela peut s'expliquer par le fait que la précision de cet oracle est la plus élevée des trois, tout en maintenant une diversité suffisante. Ceci illustre la propriété principale de l'oracle qui doit absolument privilégier sa précision au détriment de son rappel. La F-Mesure, qui accorde la même importance au rappel et à la précision, n'est donc pas un bon indicateur de la performance d'un oracle.

Enfin, malgré toutes ces tentatives, le classifieur contextualisé à l'aide d'un oracle parfait, c'est-à-dire à l'aide de la vérité terrain, reste le plus performant. Comme il est difficile de construire un oracle encore plus précis sans faire chuter le rappel, d'autres pistes pourraient être explorées. Par exemple il devrait être possible d'améliorer les performances du système final en travaillant sur d'autres aspects de la contextualisation comme l'analyse spatiale de la scène.

L'oracle 2D ainsi que ses améliorations apportent une première solution efficace au problème de constitution d'une base d'apprentissage. La suite du chapitre est consacrée à la réalisation d'un oracle 3D dont la structure, notamment la fusion des signaux, diffère quelque peu de celle utilisée par l'oracle 2D.

4.4 Oracle 3D

Cette partie va maintenant s'intéresser à la création d'un oracle 3D, la caméra doit donc être préalablement calibrée. Les gains espérés sont de deux sortes. Tout d'abord, le parcours de l'oracle dans l'image peut désormais s'effectuer en utilisant l'hypothèse du plan du sol(*cf* paragraphe 2.3.3.3). Cela permet de limiter l'espace de recherche et de gagner du temps de calcul. D'autre part cette information ouvre de nouvelles perspectives pour exploiter de manière plus précise les informations provenant de la soustraction de fond.

4.4.1 Caractéristiques de l'oracle 3D

Dans la partie 4.3, nous avons rencontré plusieurs difficultés. Le fait que la détection 2D ne contraint pas la taille des piétons, nous avait encouragé à utiliser des classifieurs de soustraction de fond et de flot optique, pour construire des modèles plus robustes ce qui a pour effet d'augmenter significativement les temps de calcul.

D'autre part la fusion des différents signaux pourrait être améliorée. En effet, les résultats fournis par chaque classifieur constituant l'oracle sont d'abord regroupés indépendamment les uns des autres. Ils servent ensuite à déterminer la présence d'un piéton. Dans notre cas, une intersection entre les différents signaux est effectuée. Cette approche présente deux inconvénients :

- Elle dépend de la méthode de regroupement appliquée à chaque signal. Une partie de l'information est perdue au moment de la fusion entre les signaux.
- Elle privilégie le signal d'apparence par rapport aux autres. Les piétons qui ne sont pas détectés par le classifieur d'apparence ne pourront pas être incorporés dans la base contextualisée alors qu'intuitivement ces exemples difficiles à classer pourraient apporter de l'information.

Dans cette partie nous proposons une approche permettant d'atténuer ces défauts. Concernant la fusion, nous l'avons fait intervenir en même temps que le regroupement, ce qui permet de travailler avec la plus grande partie de l'information des deux signaux. Nous avons aussi diminué l'importance du signal d'apparence face à la soustraction de fond. Cette dernière est désormais capable de proposer des détections.

De la même façon que pour l'oracle 2D, un schéma 4.19 présentant l'architecture complète de l'oracle 3D est disponible à la fin de cette partie (page 101).

4.4.2 Fusion des signaux : minimisation d'une fonction

Dans cet oracle, pour diminuer les temps de calcul, nous nous limitons à l'utilisation des signaux d'apparence et de soustraction de fond. De plus l'image segmentée fournie

par la soustraction est utilisée directement. Il ne reste donc plus qu'un seul classifieur dans l'oracle, celui d'apparence. Il nous reste alors à définir un algorithme de fusion en adéquation avec nos exigences.

Notre approche est inspirée du travail de [Rodriguez *et al.*, 2011]. Elle consiste à modifier l'algorithme de suppression des non maximums (*cf* paragraphe 2.3.3.4, page 39) pour y incorporer la fusion. Un des avantages de cette approche est que la soustraction de fond et le classifieur d'apparence ont un rôle plus symétrique qu'auparavant. Cela veut dire qu'il est possible de corriger le classifieur d'apparence en filtrant des fausses détections comme en 2D, mais aussi et surtout en ajoutant des piétons non détectés.

L'ensemble des boîtes testées dans l'image est indexé de 1 à n. Le classifieur générique d'apparence fournit un vecteur Θ contenant les scores de classification θ_i de toutes les boîtes. Ce vecteur est ensuite normalisé pour devenir Θ^* comme expliqué dans le paragraphe suivant. Nous disposons, par ailleurs, d'une image binaire de soustraction de fond I_{SdF}. Le but de la fusion est de produire, à partir de toutes ces entrées, un vecteur de booléens $z \in \{0,1\}^n$, dont la $i^{ème}$ composante indique si la $i^{ème}$ boîte correspond à un piéton.

Ici nous proposons de voir la fusion des signaux comme la minimisation d'une fonction E définie comme suit :

$$E(z) = -(\Theta^*)^T z + z^T W z + \alpha \times d_{\text{SdF}}(I_{\text{SdF}}, I_{\text{Modèle}}(z)) \tag{4.1}$$

Le terme $(\Theta^*)^T z$ représente la somme des scores normalisés des détections sélectionnées.

Comme dans l'algorithme classique de suppression des non maximums, dès qu'une boîte est considérée comme étant une détection, il ne peut pas y avoir d'autres détections dans un voisinage proche. Ce rôle de suppression des boites qui sont trop proches est dévolu à la matrice W. Le terme $z^T W z$ pénalise les boîtes trop proches d'une détection validée.

Le dernier terme lie l'apparence et la soustraction de fond. C'est lui qui assure la fusion entre les deux signaux. Il mesure la différence entre la segmentation de la soustraction de fond et un modèle de soustraction de fond fabriqué à partir des détections du classifieur d'apparence (voir paragraphe 4.4.3).

Le paramètre α permet de moduler les influences respectives de la classification (signal d'apparence) et de la soustraction de fond. Plus il est bas, moins la soustraction de fond a d'importance et moins elle filtre ou ajoute de détections.

À noter que minimiser les deux premiers termes de cette fonction $(-(\Theta^*)^T z + z^T W z)$ est une formalisation mathématique de l'algorithme classique de suppression des non maximums (*cf* paragraphe 2.3.3.4, page 39).

L'information de calibrage semble ne pas apparaître dans l'approche qui vient d'être décrite. En fait, l'hypothèse du plan du sol est utilisée de manière indirecte. Elle permet de connaître la taille des piétons en fonction de leur localisation dans l'image et donc de restreindre très fortement les positions et les échelles admissibles. Cette information

contraint considérablement le problème de fusion et élimine un grand nombre de faux positifs. En effet, le modèle de soustraction de fond est trop faible pour repérer de manière certaine la présence d'un piéton. En l'absence d'information sur la taille des personnes, n'importe quelle perturbation dans le signal de soustraction de fond peut conduire à la création de fausses détections même si celles-ci sont manifestement trop petites ou trop grandes. C'est d'ailleurs cette même faiblesse des réponses des signaux temporels qui nous avait conduit à construire des classifieurs de soustraction de fond et de flot optique dans le cadre de l'oracle 2D (*cf* paragraphe 4.3.1). En 3D, la connaissance de la géométrie de la scène permet de simplifier l'utilisation de la soustraction de fond.

4.4.3 Implémentation

L'oracle 3D utilise des signaux d'apparence et de soustraction de fond, les mêmes que pour l'oracle 2D. Le classifieur d'apparence est identique, seule varie la méthode de parcours dans l'image qui prend en compte le plan du sol.

Lors de la fusion des signaux d'apparence et de soustraction de fond, une étape de normalisation entre 0 et 1 de la carte de scores du classifieur d'apparence est effectuée à l'aide d'un calcul de contraste :

$$\theta^*_{t,i} = \frac{\theta_{t,i} - \min\limits_{1\leq j\leq t}\{\Theta_j\}}{\max\limits_{1\leq j\leq t}\{\Theta_j\} - \min\limits_{1\leq j\leq t}\{\Theta_j\}} \tag{4.2}$$

avec :

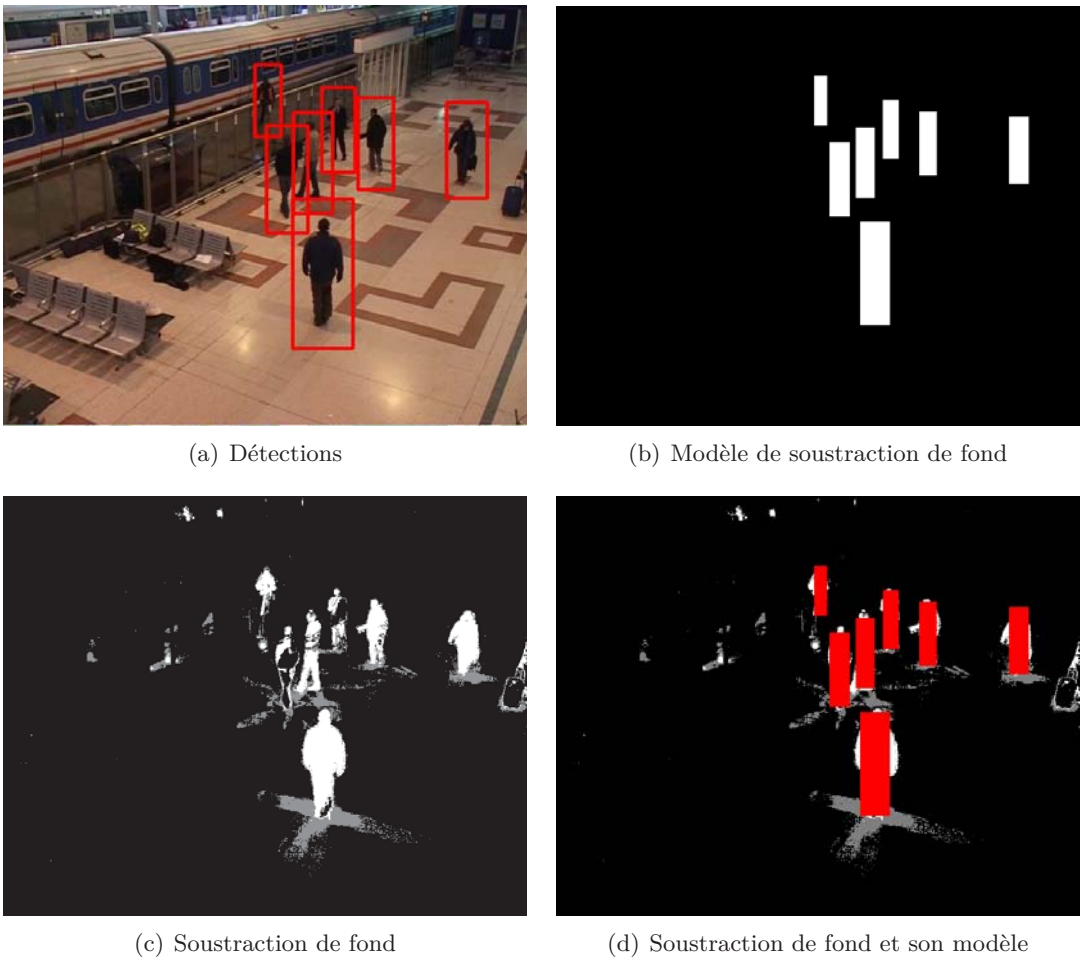
- $\theta_{t,i}$ la $i^{ème}$ composante du vecteur Θ_t, ensemble des scores de classification de l'image courante indicée par t.
- $\min\limits_{1\leq j\leq t}\{\Theta_j\}$ le score de classification minimum figurant dans tous les vecteurs Θ_j des images précédant l'image courante. De même pour le maximum.

Il n'est pas souhaitable de réaliser une normalisation, de manière indépendante, image par image. En effet, dans le cas où l'image courante ne contiendrait aucun piéton, tous les scores de classification seraient bas et relativement proches avant normalisation. Cependant, ils se retrouveraient mécaniquement étalés dans l'intervalle $[0, 1]$ après normalisation. Par contre si un piéton est présent, le fond aura des scores faibles avant et après la normalisation. Cela n'est pas acceptable car des éléments du fond se retrouveraient avec un score élevé lorsqu'il n'y a pas de piétons et avec un score faible lorsqu'une personne traverse la scène. Il est donc impératif de lisser temporellement la normalisation, pour que les scores de la scène soient stables dans le temps. Ainsi les termes maximum et minimum des scores de classification de l'équation (4.2) correspondent aux extremums non pas de l'image courante mais de toutes les images traitées précédemment indicées par j. Comme nous travaillons avec une caméra fixe, les scores sont comparables au fil de la séquence et au bout de quelques images, environ 100, les extremums ne varient presque plus.

La matrice W est symétrique et de taille $n \times n$ avec $W_{i,j} = +\infty$ si les boites i et j sont similaires et 0 sinon. Ainsi si deux boîtes i et j sont sélectionnées alors qu'elles

sont similaires, le terme E tend vers $+\infty$ ce qui est incompatible avec la minimisation. Pour fabriquer la matrice W, nous avons besoin de définir une similarité entre boîtes. Le critère de similarité choisi est, comme dans le cas 2D, celui retenu dans le challenge PASCAL (*cf* paragraphe 2.3.3.4, page 39).

Notre méthode fait également appel à un modèle de segmentation fond/forme appelé $I_{\text{Modèle}}(z)$. Il est construit à partir du vecteur z des détections. Nous considérons qu'un humain est représenté dans l'image par un rectangle qui correspond à la boîte de détection privée des marges entourant normalement le piéton. Ainsi notre modèle de soustraction de fond est une union de rectangles, remplissant chaque détection (*cf* figure 4.18). Il pourrait être affiné. Par exemple modéliser des personnes par des ellipses pourrait être plus judicieux. Néanmoins en pratique l'utilisation de rectangles est plus rapide et fournit de bons résultats.



(a) Détections (b) Modèle de soustraction de fond

(c) Soustraction de fond (d) Soustraction de fond et son modèle

FIGURE 4.18 – Modèle de soustraction de fond (b) généré grâce aux détections du classifieur d'apparence (a). La différence entre la soustraction de fond (c) et le modèle est représenté en (d).

La distance entre le modèle estimé et la segmentation réelle de soustraction de fond est :

$$d_{\text{SdF}}(I_{\text{SdF}}, I_{\text{Modèle}}(z)) = \left| \sum_{r \in Rect(z)} \sum_{i_r, j_r} [p_{\text{Modèle}}(i_r, j_r) - p_{\text{SdF}}(i_r, j_r)] \right| \tag{4.3}$$

$$= \sum_{r \in Rect(z)} [1 - \sum_{i_r, j_r} p_{\text{SdF}}(i_r, j_r)] \tag{4.4}$$

avec :

- $Rect(z)$ l'ensemble des rectangles du modèle sachant z ;
- $p(i_r, j_r)$ la valeur v du pixel de l'image du modèle ou de soustraction de fond au point (i_r, j_r) inclus dans le rectangle r ramenée à la surface de ce dernier (soit $p(i_r, j_r) = \frac{v(i_r, j_r)}{surface(r)}$).

Cette distance permet de mesurer la différence entre le modèle et la carte de soustraction de fond. Normalement, si une boîte contient un piéton, beaucoup de pixels seront allumés dans la carte de soustraction de fond. La distance avec le modèle est donc proche de 0. Un faible score d'apparence (compté négativement dans (4.1)) est donc suffisant pour faire diminuer la fonction E. Dans le cas contraire, si la soustraction de fond est peu sûre de la présence d'un objet, la distance avec le modèle sera proche de 1 et le score d'apparence devra être élevé pour valider la détection.

Soustraction de fond et apparence ont donc un rôle assez symétrique. L'un comme l'autre peuvent proposer des détections. Nous ne l'avons pas testé mais il est tout à fait possible de baser l'heuristique de fusion en privilégiant la soustraction de fond et non l'apparence. Ceci est d'ailleurs utilisé lorsque pour limiter les calculs, le classifieur d'apparence ne parcourt que les zones où la soustraction de fond suppose qu'il y a un objet.

À noter qu'il n'y a pas dans le cas de l'oracle 3D contrairement à celui 2D de restrictions sur le score du classifieur d'apparence. Toutes les boîtes sont prises en compte à condition que la soustraction de fond soit suffisamment confiante sur la présence d'un objet.

La méthode de minimisation de la fonction E est formalisée dans l'algorithme 6. En pratique, minimiser E, est une opération trop coûteuse à mettre en œuvre car ce problème d'optimisation est NP-complet. Pour trouver la valeur exacte de z qui minimise E, il faudrait être exhaustif et tester les 2^n possibilités (une boîte peut être une détection ou non). Le nombre de boîtes testées dans une scène étant de l'ordre de 10 000, une heuristique est donc nécessaire pour simplifier la résolution. L'algorithme procède par itérations successives. À l'initialisation z est le vecteur nul. Le but d'une itération est de donner la valeur 1 à une composante de z, c'est-à-dire d'ajouter une détection. Comme dans le cas standard, la détection choisie est celle qui a le plus grand score de classification. Si cette boîte permet de diminuer la valeur de E alors elle est définitivement prise en compte et une nouvelle itération est lancée. Dans le cas contraire l'algorithme s'arrête.

Le paramètre α de l'équation (4.1), est réglé arbitrairement à 1 dans nos expérimentations. Les signaux d'apparence et de soustraction de fond ont donc la même importance.

Algorithme 6: Oracle 3D : Fusion des signaux, minimisation d'une fonction

Entrées :
Soit $\Theta_t = \{\theta_{t,i}\}_{1 \leq i \leq n}$ le vecteur contenant les scores de classification de l'image t associés aux boîtes B_i définies grâce à la calibration
Soit I_{SdF} la segmentation binaire fond-forme de l'image

Sorties :
Un vecteur z contenant l'ensemble des détections après fusion

Algorithme :

Initialiser E et E_{tmp} à 0
Initialiser z et z_{tmp} à 0
Initialiser $I_{\text{Modèle}}$ à $\emptyset$
Normalisation : $\theta^*_{t,i} = \frac{\theta_{t,i} - \min\limits_{1 \leq j \leq t} \{\Theta_j\}}{\max\limits_{1 \leq j \leq t} \{\Theta_j\} - \min\limits_{1 \leq j \leq t} \{\Theta_j\}}$

répéter
- $E = E_{tmp}$ et $z = z_{tmp}$
- Sélectionner l'hypothèse disponible i qui a le plus grand score $\theta^*_{t,i}$ de classification normalisé. Soit B_i la boîte correspondante.
- Donner la valeur 1 à la $i^{ème}$ composante de z_{tmp}
- Mettre à jour le modèle de soustraction de fond $I_{\text{Modèle}}(z_{tmp})$ avec la nouvelle hypothèse
- Calculer E_{tmp} avec l'équation (4.1)
- Supprimer toutes les hypothèses similaires à B_i (revient à mettre à jour W).
- Deux hypothèses sont similaires si $sim(B_1, B_2) = \frac{Aire(B_1 \cap B_2)}{Aire(B_1 \cup B_2)} > 0,5$

jusqu'à $E_{tmp} > E$;

Comme annoncé la figure 4.19 résume l'architecture de l'oracle 3D.

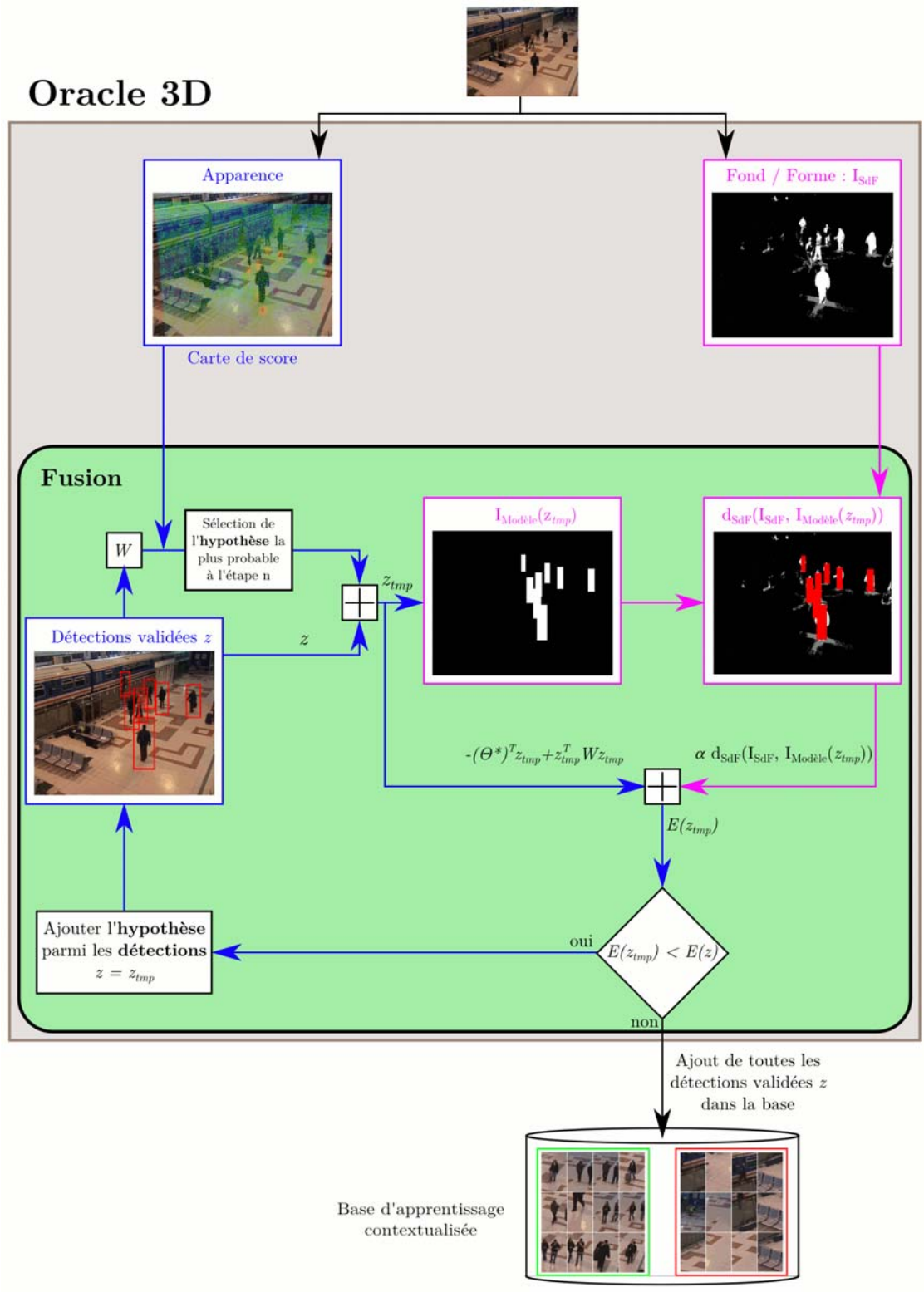


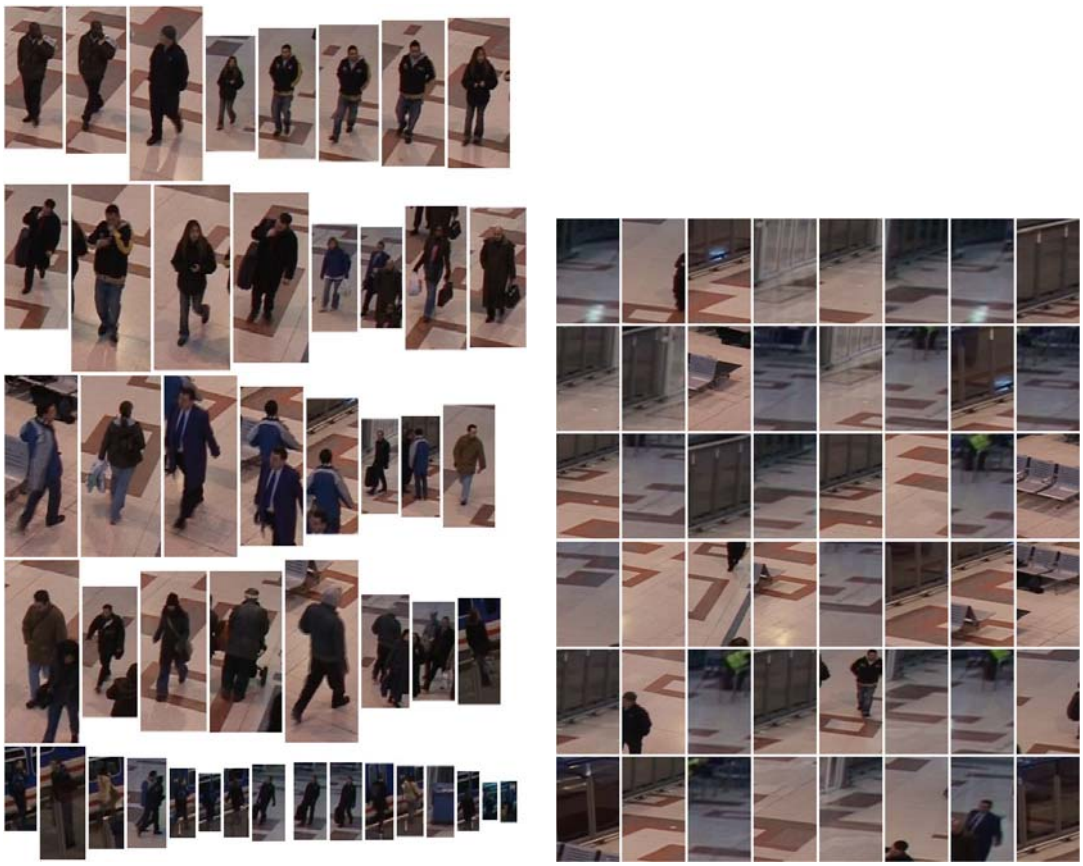
FIGURE 4.19 – Schéma complet de l'oracle 3D. Il présente l'architecture globale de l'oracle 3D et la procédure de construction des exemples positifs de la base contextualisée.

4.4.4 Validations expérimentales 3D

Nous présentons ici les résultats, issus de nos expérimentations avec l'oracle 3D, sur différentes séquences : PETS 2006 et ViCoMo 2. Le protocole utilisé est le même que celui employé lors de l'évaluation des oracles 2D.

4.4.4.1 PETS 2006

La première vidéo testée est PETS 2006. La figure 4.20 présente un échantillon des exemples collectés.



(a) Exemples positifs (b) Exemples négatifs

FIGURE 4.20 – Exemples choisis provenant de la base d'apprentissage du détecteur final pour PETS 2006

Comparé à l'oracle 2D, la première remarque est la plus grande diversité des exemples retenus. En effet cet oracle est capable de bien détecter la plupart des personnes qui sont près des trains de l'autre côté du mur de vitres (la dernière ligne de la figure 4.20(a) en présente un échantillon). Ceci est en grande partie dû à la manière dont les signaux sont fusionnés dans les deux oracles. L'oracle 2D est très restrictif car tout ce qui n'a pas été détecté par le classifieur d'apparence est perdu. En revanche, dans le cas 3D, comme la soustraction de fond contribue presque autant à la détection que l'apparence, elle permet d'augmenter le rappel. Évidemment cela fonctionne d'autant mieux que les

objets mobiles de la scène sont principalement des piétons. D'autre part, comme prévu, il n'y a plus de problème d'échelle des exemples, la taille de l'imagette est bien adaptée à la taille des piétons.

Comme l'algorithme du mean shift a été remplacé par une version modifiée de l'algorithme de suppressions des non maximums et n'est plus utilisé pour regrouper les détections, la boîte obtenue en sortie de l'oracle correspond exactement à une boîte testée par le détecteur. Cela permet de ne plus manipuler les imagettes des exemples. Nous nous sommes en effet aperçu que le simple fait de redimensionner les exemples pouvait dans certains cas conduire à une dégradation significative des performances (de 5 à 10% du rappel pour une même précision). Les seules erreurs qui subsistent dans la base sont donc quelques faux positifs et des problèmes de position avec certains piétons qui sont mal centrés.

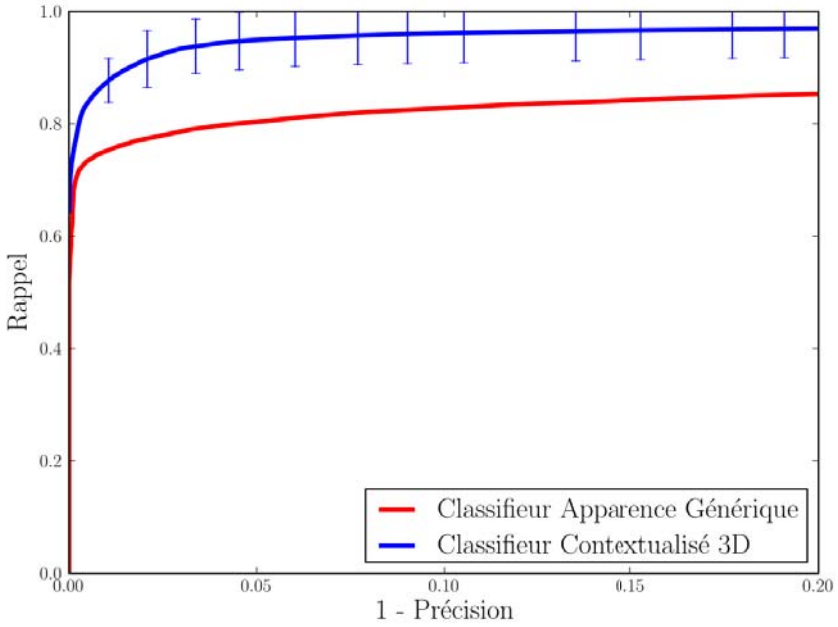


FIGURE 4.21 – Courbes précision-rappel sur la séquence PETS 2006 pour le classifieur générique d'apparence constituant l'oracle 3D, ainsi que le classifieur contextualisé 3D

La figure 4.21 présente les courbes précision-rappel obtenues avec le classifieur générique d'apparence utilisé par l'oracle ainsi que le classifieur contextualisé. Cinq classifieurs contextualisés ont été entraînés à partir de la même base fabriquée par l'oracle. La courbe bleue présente la moyenne et l'erreur à 2 σ près de ces expériences. Même si le passage en 3D augmente sensiblement les performances du système générique, la contextualisation apporte encore un net gain de performance.

La table 4.5 répertorie différents points de fonctionnement du classifieur générique de l'oracle et du classifieur contextualisé. Tous ces points ont été calculés sur la séquence de test et non celle d'apprentissage. Pour le classifieur d'apparence générique de l'oracle, nous présentons deux seuils. Le premier correspond au réglage par défaut dans l'oracle, tandis que le second correspond aux performances optimales, c'est-à-dire au seuil qui

maximise la F-Mesure.

La table 4.5 et les courbes 4.21 montrent, comme déjà expliqué dans le cas 2D, que les performances du classifieur générique ont déjà un niveau relativement bon. Encore une fois, ceci s'explique par le fait que le point de vue de cette séquence est relativement proche de celui de la base générique. Par contre et comme attendu par construction, le rappel de l'oracle est légèrement sacrifié par rapport à celui du classifieur générique, au profit de sa précision qui dépasse les 99%. Néanmoins, comme déjà évoqué lorsque nous avons étudié les exemples en sortie de l'oracle, le rappel de la version 3D est plus élevé que son homologue 2D pour sensiblement la même précision. À noter que toutes les statistiques sur l'oracle ont été obtenues après regroupement des boîtes, ce qui peut fausser les comparaisons avec les autres classifieurs, notamment pour la précision. Enfin, le classifieur contextualisé qui a été entraîné sur la base construite par l'oracle est capable d'avoir un rappel élevé tout en gardant une bonne précision. Il est donc meilleur que le générique.

TABLE 4.5 – Caractéristiques des classifieurs 3D sur la séquence PETS 2006

	Rappel	Précision	F-Mesure
Apparence oracle (seuil 0)	0,87	0,66	0,75
Apparence (seuil optimal)	0,80	0,96	0,87
Oracle	0,66	**0,99**	0,79
Classifieur contextualisé	**0,94**	0,97	**0,95**

À titre de comparaison, la table 4.6 reproduit les principaux résultats obtenus en 2D sur cette même séquence.

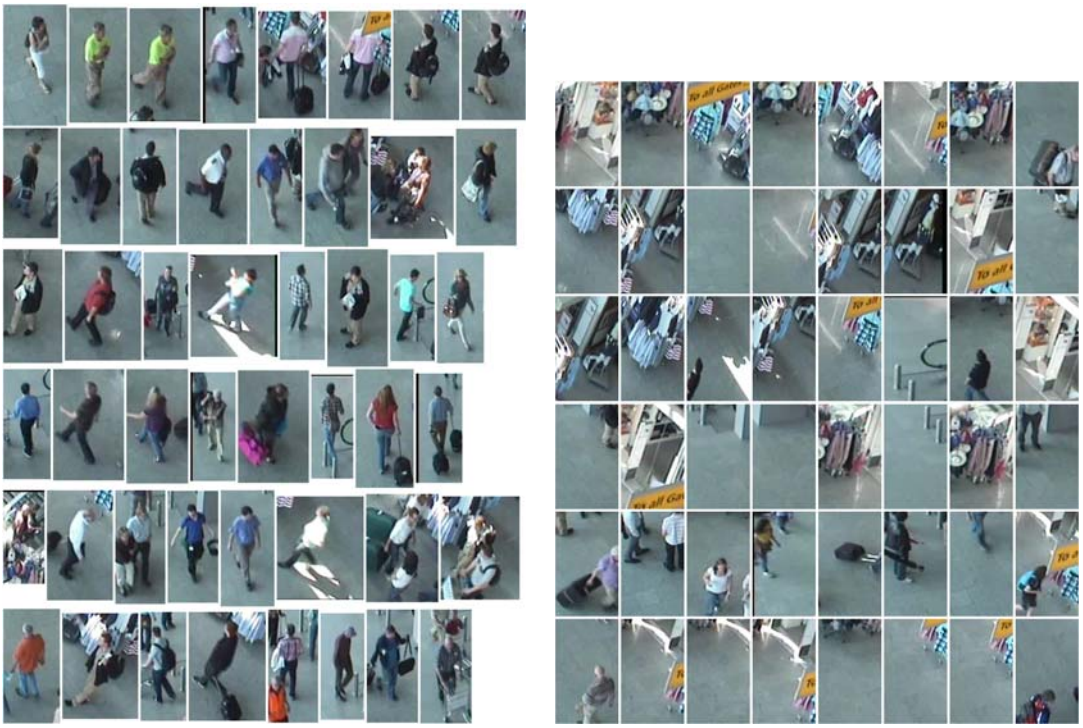
TABLE 4.6 – Caractéristiques des détecteurs 2D sur la séquence PETS 2006

	Rappel	Précision	F-Mesure
Apparence	0,71	0,66	0,69
Oracle	0,49	**0,99**	0,65
Classifieur contextualisé	**0,85**	0,90	**0,87**

4.4.4.2 ViCoMo 2

Cette séquence contient environ 45 000 images. Les 5 000 premières sont annotées et servent à la validation des classifieurs et de l'oracle En conséquence, ce dernier collecte les informations sur les images allant de 10 000 à 45 000, pour fabriquer la base d'apprentissage. Ici la perspective est forte et les piétons peuvent avoir une orientation importante par rapport à la verticale.

La figure 4.22 présente un échantillon des exemples positifs et négatifs collectés par l'oracle. Comme pour l'expérience sur PETS 2006, nous avons entraîné cinq classifieurs



(a) Exemples positifs (b) Exemples négatifs

FIGURE 4.22 – Exemples choisis provenant de la base d'apprentissage du détecteur final pour ViCoMo 2

contextualisés sur la même base. La figure 4.23 présente la moyenne et la variance des résultats. Enfin la table 4.7 présente quelques statistiques pour les seuils maximisant la F-Mesure. Encore une fois, celles de l'oracle sont réalisées après regroupement.

TABLE 4.7 – Caractéristiques des classifieurs sur la séquence ViCoMo2

	Rappel	Précision	F-Mesure
Apparence oracle (seuil 0)	0,85	$6{,}1.10^{-3}$	$1{,}2.10^{-2}$
Apparence (seuil optimal)	0,33	0,81	0,46
Oracle	0,60	**0,97**	0,74
Classifieur contextualisé	0,77	0,84	**0,80**

Encore une fois, la contextualisation améliore sensiblement les résultats. Le classifieur contextualisé est celui qui a la meilleure F-Mesure et le meilleur rappel. Néanmoins, comme précédemment, la variance des résultats est assez importante.

D'autre part, pour les précisions élevées, il y a une baisse des performances. Cela peut éventuellement s'expliquer par certaines erreurs commises par l'oracle. En effet, il y a en haut de l'image un escalier mécanique. Les piétons qui l'empruntent activent donc la soustraction de fond. De plus, des vêtements exposés juste en dessous dans l'image ressemblent à des piétons. Le classifieur d'apparence se trompe et donne à cette zone

un score élevé. Le manque de précision dans la soustraction de fond cumulé aux erreurs du classifieur d'apparence introduit des erreurs dans la base. Ces dernières sont ensuite apprises par le classifieur contextualisé ce qui a pour conséquence de renforcer le score des vêtements. Ces faux positifs ont donc un score plus important que de nombreux piétons. Lors de la construction de la courbe, les seuils de détections doivent donc augmenter fortement pour que la précision du classifieur s'améliore. Comme de nombreux piétons ont un score plus faible que les erreurs, ils ne sont plus détectés à ces seuils ce qui conduit à une baisse du rappel.

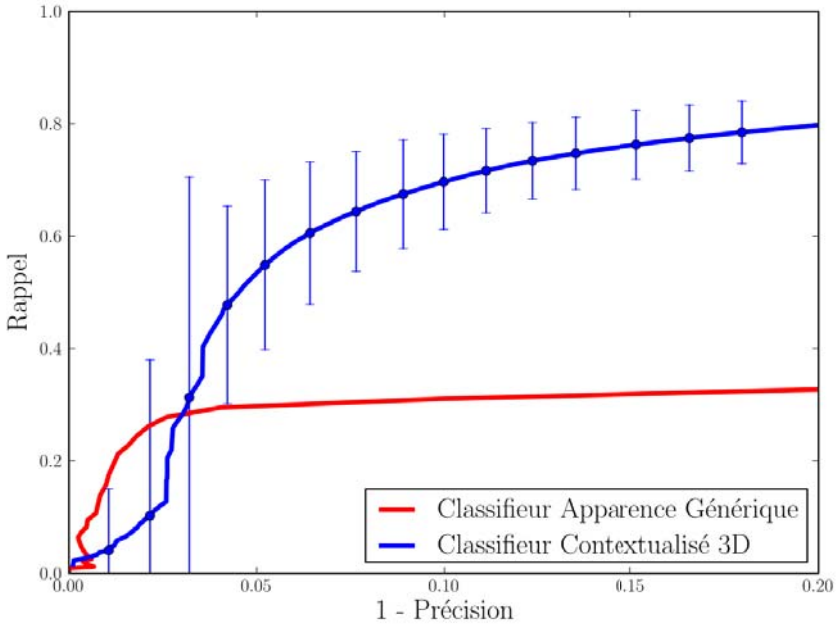


FIGURE 4.23 – Courbes précision-rappel sur la séquence ViCoMo 2 pour le classifieur générique d'apparence constituant l'oracle, ainsi que pour le classifieur contextualisé

4.5 Conclusion

Un oracle est un maillon essentiel pour réaliser la contextualisation d'un détecteur automatiquement. Il a pour rôle d'extraire un ensemble d'exemples positifs et négatifs d'une vidéo avec une grande précision.

Dans ce chapitre, des oracles 2D et 3D on été élaborés. L'oracle 2D classique a été publié dans [Chesnais *et al.*, 2012b,a], tandis que l'oracle 3D a été publié dans [Chesnais *et al.*, 2013]. L'avantage d'un oracle 2D est qu'il peut s'adapter à n'importe quelle caméra existante alors que pour bénéficier d'un oracle 3D une étape supplémentaire et coûteuse de calibration est nécessaire. Tous les oracles présentés dans nos travaux sont basés sur une combinaison de différentes briques comprenant la classification générique et des méthodes temporelles comme la soustraction de fond, le flot optique ou le tracking.

En 2D, le problème de la détection est moins contraint qu'en 3D car la taille et

l'orientation d'un piéton, en un point donné de l'image, ne sont pas connues. Il faut donc tester de nombreuses hypothèses lors de la détection. Un modèle de représentation des piétons trop faible engendre alors de nombreux faux positifs qu'il est difficile de filtrer correctement. Pour que les modèles de soustraction de fond et de flot optique soient plus robustes, il a été décidé d'entraîner un classifieur par signal, à l'aide de deux bases d'apprentissage génériques différentes. Ces deux classifieurs viennent ensuite confirmer les sorties du classifieur d'apparence. L'oracle 2D classique construit une base d'apprentissage contextualisée à l'aide de l'intersection des réponses des trois classifieurs après regroupement des boîtes. Deux améliorations de l'oracles 2D ont été proposées. L'une augmente le rappel de l'oracle grâce à un module de tracking. La seconde adapte localement les seuils de détection du classifieur d'apparence à la scène considérée pour réduire le nombre de faux positifs.

En 3D, le problème de la détection est mieux contraint. Contrairement au cas 2D où beaucoup d'hypothèses doivent être testées, ici seules les portions de l'image intéressantes sont examinées. Les faux positifs sont moins nombreux et il est donc possible de simplifier le modèle utilisé pour la soustraction de fond. Grâce à ce changement, la fusion des signaux, réalisée par le biais de la minimisation d'une fonction, évite de privilégier un signal par rapport à un autre et permet de réduire les temps de calcul.

Les performances atteintes par les classifieurs contextualisés 2D ou 3D sont nettement supérieures à celles du classifieur générique et se rapprochent parfois de celles du classifieur contextualisé manuellement. D'autre part, comme attendu, les expériences ont montré qu'un oracle ne peut être performant que si sa précision est élevée. En 2D, l'oracle le plus précis et donc celui qui permet la meilleure contextualisation repose sur les seuils adaptatifs.

Après avoir comparé les performances de chacun des oracles, nous pouvons logiquement conclure que l'utilisation d'un oracle 3D permet d'obtenir de meilleurs résultats, le problème de détection étant mieux contraint dans ce cas. Cet oracle doit donc être privilégié dès que les informations de calibration sont disponibles.

À noter pour finir que l'oracle se contente de rechercher des exemples. Il n'a aucune connaissance *a priori* de la pertinence de ceux-ci au moment de l'apprentissage contextualisé. En effet comme il travaille sur une vidéo, de nombreux exemples risquent d'être similaires puisque provenant d'images proches temporellement et il est possible que cela perturbe l'apprentissage contextualisé. Le choix d'exemples pertinents par rapport à la base en construction ne dépend pas de l'oracle. Seule compte la qualité de ses détections et donc sa précision et dans une moindre mesure son rappel. L'étape de sélection des exemples sera évoquée dans le chapitre 5 dans lequel plusieurs stratégies seront développées. Le but de cette sélection est d'améliorer les performances du détecteur contextualisé et si possible de réduire la variance des résultats observés dans ce chapitre.

Partie II : Travaux réalisés

5

Stratégies de sélection des exemples issus de l'oracle

Sommaire

5.1 Introduction 110
5.1.1 Considérations sur les exemples fournis par l'oracle 110
5.1.1.1 Vidéosurveillance 110
5.1.1.2 Exemples difficiles et exemples rares 110
5.1.2 Problèmes rencontrés 110
5.1.2.1 Choix des exemples 110
5.1.2.2 Mesures des performances 111
5.1.3 Objectifs 111
5.2 Sélection aléatoire 112
5.3 Sélection par score, par apparence et par suivi 117
5.3.1 Exemples négatifs, sélection par score : bootstrapping 117
5.3.2 Exemples positifs 119
5.3.2.1 Sélection par apparence : clustering 119
5.3.2.2 Sélection par suivi 121
5.4 Sélection par contraintes géométriques et spatiales 124
5.4.1 Intérêts d'un découpage par régions 124
5.4.2 Création des régions 126
5.4.2.1 Superpixel 126
5.4.2.2 Application à la création des régions 128
5.4.3 Validations expérimentales 132
5.4.3.1 Résultats par région 132
5.4.3.2 Résultats globaux 133
5.5 Conclusion 135

5.1 Introduction

Grâce aux oracles présentés dans le chapitre précédent, une base d'apprentissage, contenant des piétons en provenance de la scène, a été construite. Le but de ce chapitre est de fabriquer le classifieur contextualisé le plus performant possible à partir de ces exemples.

5.1.1 Considérations sur les exemples fournis par l'oracle

5.1.1.1 Vidéosurveillance

La base constituée par l'oracle est bien souvent imparfaite. Elle contient inévitablement quelques erreurs de label et de position. De plus, de nombreux exemples sont proches voire quasi-similaires. En effet, dans le cas de la vidéosurveillance, la caméra est fixe. Cela signifie que, d'une part, les exemples négatifs sont tous très proches car provenant tous du fond de la même image (aux conditions d'éclairage près) et que d'autre part les exemples positifs comportent plusieurs instances de la même personne. Avoir des milliers d'exemples qui n'apportent pas d'information particulière n'est pas souhaitable. Cela augmente les temps de calcul (pouvant atteindre une à deux heures lorsque la centaine de milliers d'exemples est franchie) et la mémoire pour stocker les données. Enfin, les performances de l'apprentissage peuvent être dégradées par ce manque de diversité.

5.1.1.2 Exemples difficiles et exemples rares

En apprentissage statistique, certains exemples sont plus intéressants que d'autres. Il s'agit principalement des exemples difficiles, c'est-à-dire ceux que le classifieur a du mal à traiter correctement. Ces exemples sont importants car ils sont généralement proches de la séparation des classes et peuvent donc apporter de l'information sur cette frontière.

Les exemples rares sont des exemples qui ont statistiquement peu de chances de se produire. Il peut s'agir d'événements anormaux comme un piéton qui entre dans une zone fermée au public ou peu accessible. Tout comme les exemples difficiles, ces observations sont potentiellement utiles. En effet, elles peuvent renseigner sur la répartition des exemples et donc sur la séparation des classes dans une portion de l'espace où il y a peu de données.

5.1.2 Problèmes rencontrés

5.1.2.1 Choix des exemples

Le choix des exemples est un problème très délicat en apprentissage. Il ne se pose pas uniquement lors de la contextualisation d'un classifieur mais est récurrent. Néanmoins dans le cas d'un apprentissage générique, comme il n'est pas toujours facile de savoir à l'avance dans quelles conditions le détecteur va être employé, il est d'usage de prendre la base d'apprentissage la plus volumineuse et diverse possible. L'objectif est de couvrir un maximum de cas pour augmenter la variété des exemples et obtenir un classifieur avec de meilleures performances.

Si les seuls exemples choisis sont issus de la scène, la variété de la base n'est pas forcément très élevée. Dans le cas de la contextualisation, est-il toujours nécessaire de prendre tous les exemples labellisés par l'oracle puisque beaucoup sont semblables et n'apportent pas plus d'informations ? De plus, une fois le nombre d'exemples choisi, quel est le ratio optimal entre les exemples positifs et les exemples négatifs ? Ce chapitre va tenter de répondre à ces questions.

D'autre part, dans le cadre de la contextualisation la prise en compte des exemples difficiles et/ou rares est compliquée. En effet, contrairement au cas idéal dans lequel tous les exemples sont corrects, la base est bruitée. Cela signifie que certains exemples peuvent être considérés comme difficiles alors qu'ils sont tout simplement faux. En outre, l'échantillonnage réalisé par l'oracle sur les observations de la scène n'a pas forcément été uniforme. Concrètement, cela signifie qu'un événement rare peut être surreprésenté ou au contraire ne pas être représenté du tout par rapport à d'autres phénomènes plus courants mais que l'oracle a eu du mal à détecter. C'est le cas notamment dans PETS 2006. Les piétons proches du train, peu fréquents, ne sont pas détectés par l'oracle 2D. Insister lors de l'apprentissage sur des exemples faux ou sur des exemples rares au détriment d'autres plus courants peut conduire à une dégradation des performances.

5.1.2.2 Mesures des performances

Une autre difficulté est la mesure des performances. Pour déterminer si une stratégie de sélection des exemples est efficace, il faut étudier les performances du classifieur contextualisé. Une meilleure base et une meilleure stratégie de sélection sont sensées produire un meilleur classifieur. Mais une amélioration dans la sélection des exemples peut tout simplement être confondue avec le bruit de la mesure lors de l'évaluation du classifieur contextualisé. Il est donc important de réaliser chaque expérience plusieurs fois afin d'avoir une estimation plus précise de l'apport d'une stratégie par rapport à une autre.

5.1.3 Objectifs

L'objectif de cette partie est d'explorer plusieurs stratégies pour filtrer efficacement la base d'apprentissage afin d'obtenir le meilleur apprentissage contextualisé possible.

Cela pose la question : qu'est-ce qu'un exemple pertinent par rapport à la base déjà construite ?

La partie 5.2 va revenir sur la stratégie de sélection aléatoire des exemples. Le but est de trouver le nombre idéal d'exemples lors d'un apprentissage contextualisé et de justifier les quantités utilisées dans le chapitre précédent. D'autre part, nous verrons que de nombreux classifieurs ont des performances proches, ce qui signifie que le processus de contextualisation semble saturer. Fort de ces constatations, nous tenterons d'aller plus loin et de comprendre d'où provient cette saturation. Dans un premier temps, nous expérimentons d'autres stratégies de sélection des exemples négatifs et positifs. Puis dans un deuxième temps, nous proposons un apprentissage contextualisé par régions (partie 5.4). Son principal intérêt est de simplifier le problème de l'apprentissage en diminuant la variablilité des exemples dans les classes positives et négatives. En outre cela permet de complexifier le modèle de représentation des piétons, sans augmenter

les temps de calcul lors de la phase de détection, et de lutter contre le phénomène de saturation.

5.2 Sélection aléatoire

Dans le chapitre précédent, la sélection des exemples a consisté à tirer aléatoirement et sans remise un certain nombre d'exemples parmi ceux rassemblés par l'oracle. Environ 2 000 exemples positifs et 8 000 exemples négatifs servaient à l'apprentissage. Cette approche, très simple à mettre en œuvre, donne une première approximation de la quantité d'information contenue dans la base contextualisée.

Dans le but d'essayer de minimiser la quantité de données utilisées et puisque les exemples proviennent tous de la même scène et sont donc similaires, il est possible qu'un petit nombre d'entre eux suffise à atteindre de bonnes performances. Si oui combien d'exemples sont-ils nécessaires à un bon apprentissage et quel est le ratio optimal entre le nombre d'exemples positifs et le nombre d'exemples négatifs ?

Pour répondre à ces questions, plusieurs expériences ont été réalisées. La séquence choisie est ViCoMo 2 car il s'agit de la scène pour laquelle le classifieur contextualisé a les moins bonnes performances. L'oracle 3D y extrait 21 181 exemples positifs et 139 425 exemples négatifs.

À partir de cette base, plusieurs apprentissages sont réalisés. Le protocole est identique à celui des chapitres précédents. Seul change, à chaque fois, le nombre d'exemples sélectionnés et appris par le classifieur contextualisé afin de couvrir différents ratios possibles entre exemples positifs et exemples négatifs.

La figure 5.2 représente les trente-cinq meilleurs classifieurs obtenus, c'est-à-dire ceux qui ont la meilleure F-Mesure. Une courbe correspond à la moyenne des classifieurs obtenus avec ce nombre d'exemples. Chaque expérience ayant été réalisée cinq fois. De la même façon, la figure 5.3 indique les trente-cinq classifieurs les moins performants.

Il est possible de faire plusieurs remarques suite à ces expériences. Tout d'abord, l'allure générale du faisceau de courbes montre que le nombre d'exemples lors d'un entraînement influe fortement sur les performances d'un classifieur pour les précisions élevées mais a moins d'importance pour les autres. Pour une précision de 95%, les meilleurs classifieurs atteignent 60% de rappel contre 30% pour les moins bons, tandis qu'à 80% de précision, les meilleurs classifieurs arrivent à un rappel compris entre 75% et 80% contre 65% à 70% pour les plus faibles. Pour les niveaux de précision atteints en pratique par le classifieur contextualisé (entre 80% et 85%), aucune courbe n'est significativement meilleure que les autres. Le choix du nombre d'exemples ne semble donc pas très sensible.

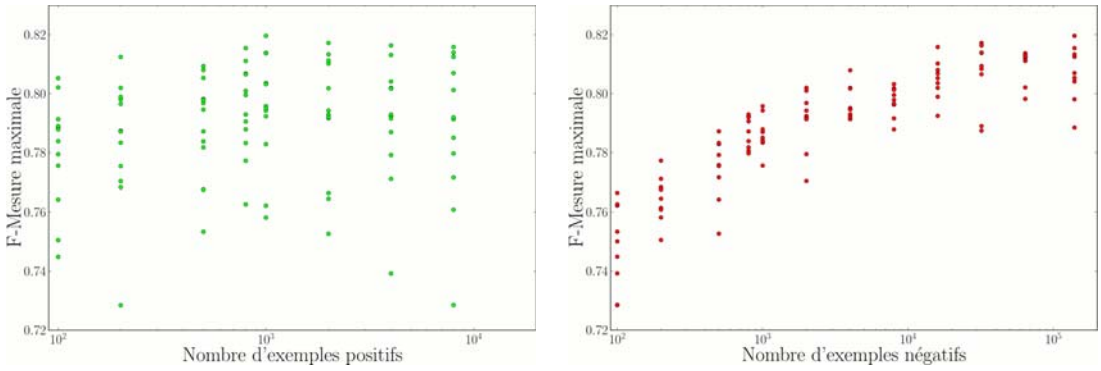
Le tableau de classement des meilleurs classifieurs indique que même s'il est bien situé (onzième position), le classifieur entraîné avec tous les exemples n'est pas le meilleur. Plus d'exemples ne signifie donc pas obligatoirement de meilleures performances.

Le même tableau montre qu'à quelques exceptions près, les meilleurs classifieurs ont été entraînés sur beaucoup d'exemples négatifs, souvent même le maximum disponible dans la base, tandis que les plus mauvais classifieurs n'en ont presque pas appris.

Dans cette scène, environ 10 000 hypothèses (c'est-à-dire des positions) sont testées par image. Comme les exemples négatifs sont extraits à partir de ces hypothèses, cela signifie qu'il y en a au maximum 10 000 par image. De plus, la scène étant quasiment fixe, les images successives sont très proches et les exemples sélectionnés aussi. En outre, l'image est échantillonnée de manière dense par les hypothèses. Au final, deux exemples pris dans deux images et à des positions différentes peuvent donc être très similaires. Intuitivement, l'ordre de grandeur pour le nombre d'exemples négatifs différents est de 10 000 pour la séquence. Il semble donc inutile et contradictoire d'en choisir beaucoup plus. Pourtant les meilleurs classifieurs sont ceux qui en utilisent jusqu'à dix fois plus lors de l'apprentissage.

L'importance des exemples positifs est plus contrastée. Il n'y a pas comme pour les exemples négatifs de séparation nette entre les classifieurs qui ont appris beaucoup d'exemples positifs et les autres. Certains classifieurs sont très bons avec seulement 100 exemples positifs. D'autres avec 20 000 exemples positifs sont peu performants.

La figure 5.1 confirme cela. Elle représente la répartition de la F-Mesure maximale d'un classifieur en fonction des quantités d'exemples positifs et négatifs utilisées lors de l'entraînement. Le nombre d'exemples positifs ne semble pas avoir d'influence sur les performances du classifieurs. Même les classifieurs entraînés avec 100 exemples obtiennent des bonnes performances. En revanche, les classifieurs ayant été construits avec une base d'apprentissage contenant moins de 4 000 exemples négatifs ont des performances plus faibles que les autres. Au delà de 4 000 exemples négatifs, la F-Mesure continue d'augmenter mais de manière plus limitée.



(a) Exemples positifs (b) Exemples négatifs

FIGURE 5.1 – Répartition de la F-Mesure maximale d'un classifieur en fonction des quantités d'exemples positifs et négatifs utilisées lors de l'entraînement (échelle semi-logarithmique).

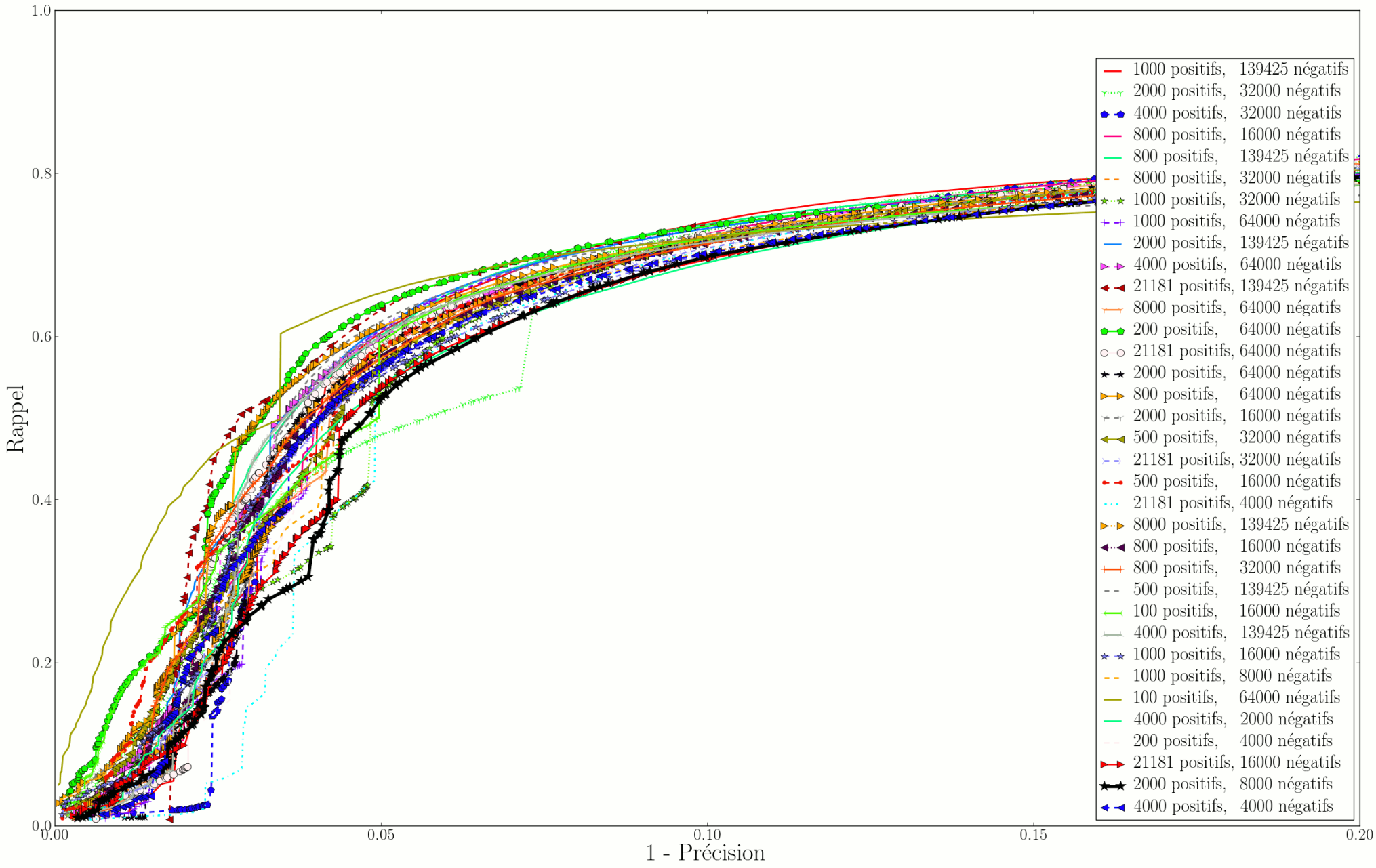


FIGURE 5.2 – Courbes précision-rappel pour les 35 meilleurs classifieurs contextualisés obtenus sur la séquence ViCoMo 2. Plus un classifieur est haut dans le tableau plus il est performant. (classement selon le critère de la F-Mesure)

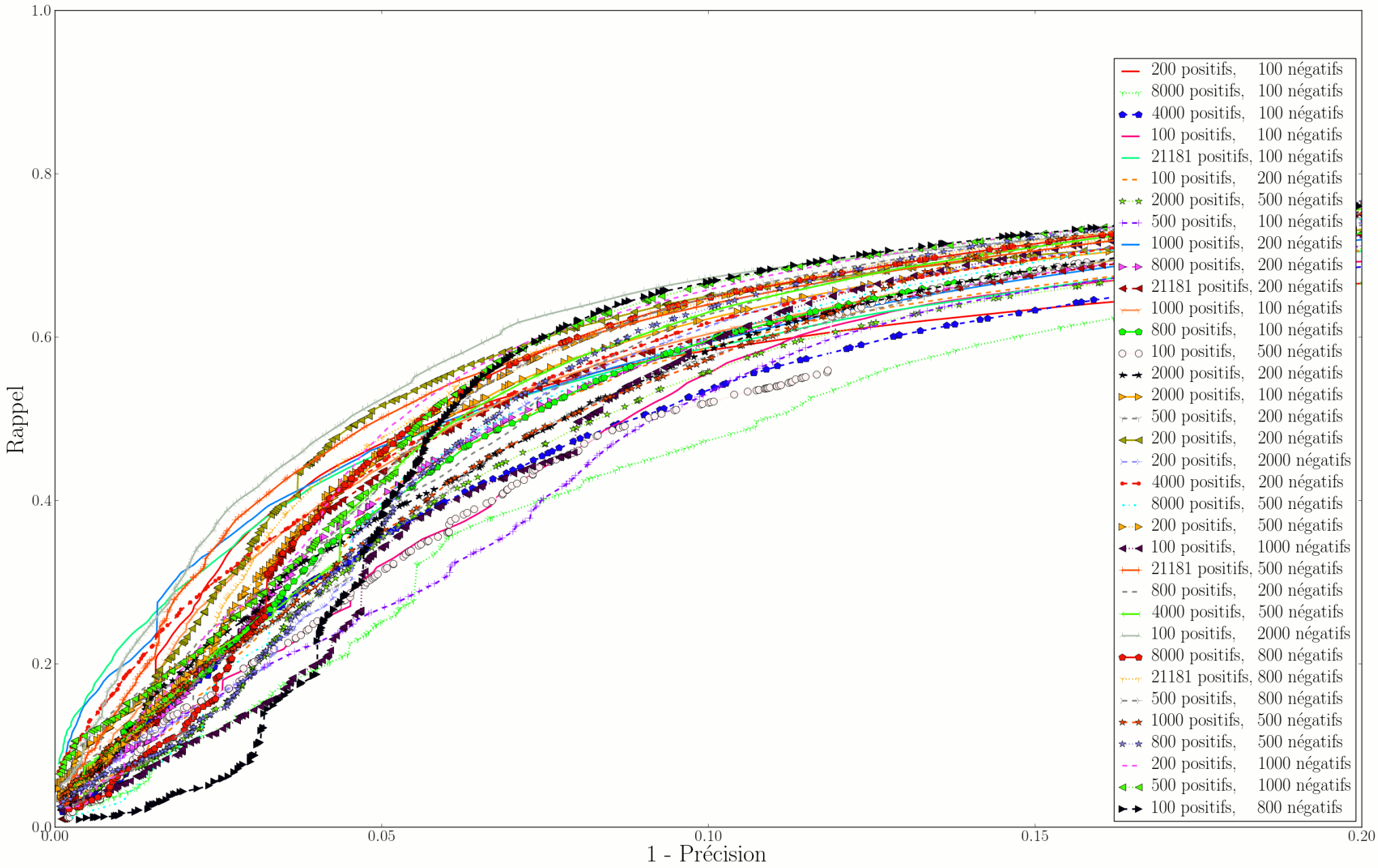


FIGURE 5.3 – Courbes précision-rappel pour les 35 classifieurs contextualisés les moins performants obtenus sur la séquence ViCoMo 2. Plus un classifieur est haut dans le tableau moins il est performant. (classement selon le critère de la F-Mesure)

Il est maintenant possible de justifier le choix de travailler, durant cette thèse, avec 2 000 exemples positifs et 8 000 exemples négatifs lors des apprentissages contextualisés. Les temps de calcul imposent un nombre d'exemples restreints. Il n'est pas envisageable lors de la mise en place de notre système de monopoliser pendant plusieurs heures un ordinateur moderne pour entraîner un classifieur dédié à une caméra. Comme déjà expliqué, la scène contient environ 10 000 positions d'exemples potentiels. 10 000 exemples pour un apprentissage semble donc être un bon compromis ne nécessitant pas plus de quelques minutes de calcul (environ 5 minutes).

La figure 5.4 présente les cinq meilleurs classifieurs entraînés avec 10 000 exemples au plus. Globalement, ils offrent tous des performances très proches. Puisqu'il est important d'entraîner avec le maximum d'exemples négatifs, nous avons décidé de travailler avec 8 000 d'entre eux. Dans cette catégorie, les deux meilleurs classifieurs ont appris avec 1 000 ou 2 000 exemples positifs. Les variances reportées sur les courbes sont plus faibles pour le classifieur ayant appris 2 000 exemples positifs, ce qui veut dire que les performances sont plus facilement reproductibles d'un entraînement à l'autre.

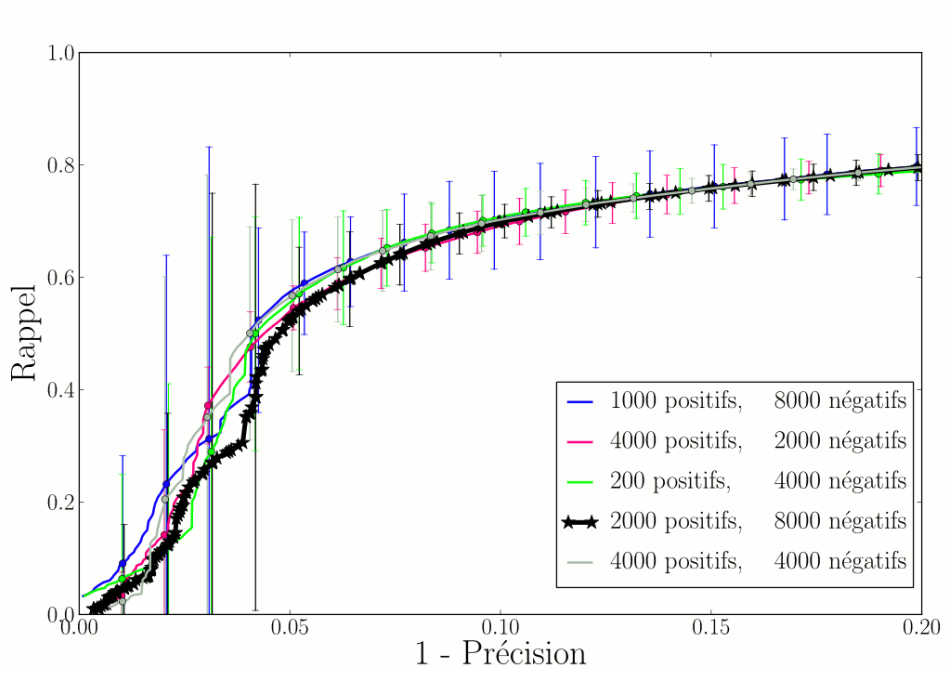


FIGURE 5.4 – Courbes précision-rappel pour les 5 meilleurs classifieurs contextualisés obtenus sur la séquence ViCoMo 2 appris avec au plus 10 000 exemples. Plus un classifieur est haut dans la légende moins il est performant.

Enfin la plupart des classifieurs offrent des performances similaires et saturent vers 70% de rappel pour 80% de précision. Dans la suite du chapitre, plusieurs expérimentations vont être réalisées pour améliorer ces résultats et tenter de dépasser cette saturation.

5.3 Sélection par score, par apparence et par suivi

L'objectif de cette partie est d'explorer plusieurs stratégies pour sélectionner les exemples. Le but est soit de diminuer la variance dans les résultats, soit d'améliorer les performances globales. Nous démarrons ce paragraphe avec l'étude du choix des exemples négatifs car, suite aux expérimentations réalisées dans la section précédente, ils semblent plus importants que les exemples positifs.

5.3.1 Exemples négatifs, sélection par score : bootstrapping

La sélection aléatoire des exemples a prouvé que les négatifs jouent un rôle prépondérant dans l'apprentissage. Cependant les classifieurs offrant les meilleures performances requièrent beaucoup d'exemples, ce qui complexifie considérablement le processus d'apprentissage. L'objectif de ce paragraphe est de savoir s'il est possible de réduire le nombre d'exemples négatifs tout en gardant de bonnes performances.

Une technique fréquemment employée lors de l'apprentissage est le *bootstrapping* ([Dalal et Triggs, 2005; Walk *et al.*, 2010; Dollár *et al.*, 2009]). Son but est de rendre le détecteur plus robuste en fournissant des exemples négatifs difficiles lors de l'apprentissage afin de mieux modéliser la séparation entre les classes. C'est une méthode d'apprentissage itérative qui consiste à entraîner plusieurs fois (entre cinq et dix fois) le classifieur. À l'initialisation, une base d'apprentissage, a_0 est créée en tirant aléatoirement des exemples positifs et négatifs parmi ceux qui sont fournis par l'oracle (base d'apprentissage A). Un classifieur h_0 est ensuite entraîné à l'aide de a_0. À l'étape i, tous les exemples négatifs de A sont traités par le classifieur h_{i-1} qui leur attribue un score à chacun. Les exemples négatifs ayant les scores les plus élevés sont incorporés à a_{i-1} pour former a_i. Ces exemples négatifs correspondent donc aux plus difficiles de A. Un nouveau classifieur h_i est alors entraîné avec a_i. Ainsi les classifieurs sont entraînés pour mieux appréhender les exemples les plus difficiles. Les exemples positifs restent fixes durant toute la procédure.

Pour étudier l'influence du *bootstrapping*, nous avons réalisé l'expérience suivante sur la séquence ViCoMo 2. À l'initialisation, cinq classifieurs sont entraînés à l'aide de cinq bases différentes nommées $a_{0,j}$ (avec $j \in \{0..4\}$) contenant chacune 2 000 exemples positifs et 2 000 exemples négatifs choisis aléatoirement. À l'issue de l'étape $i-1$ et pour chaque classifieur $h_{i-1,j}$, les 1 000 exemples négatifs ayant le score de classification le plus élevé selon $h_{i-1,j}$ parmi les exemples négatifs de A sont ajoutés dans la base $a_{i-1,j}$ Un nouveau classifieur $h_{i,j}$ est ensuite entraîné avec $a_{i,j}$. Il y a donc bien à chaque itération, cinq classifieurs différents et indépendants. Six itérations de *bootstrapping* sont réalisées successivement pour obtenir des classifieurs ayant appris 2 000 exemples positifs et 8 000 exemples négatifs.

Les résultats sont présentés sur la figure 5.5. Sur cette dernière sont aussi représentés deux classifieurs dont la base a été entièrement sélectionnée de manière aléatoire. L'une contient 2 000 positifs et 8 000 négatifs comme les classifieurs en fin de *bootstrapping* et l'autre contient 1 000 positifs et 139 425 négatifs ce qui correspond aux meilleurs classifieurs entraînés sur cette séquence.

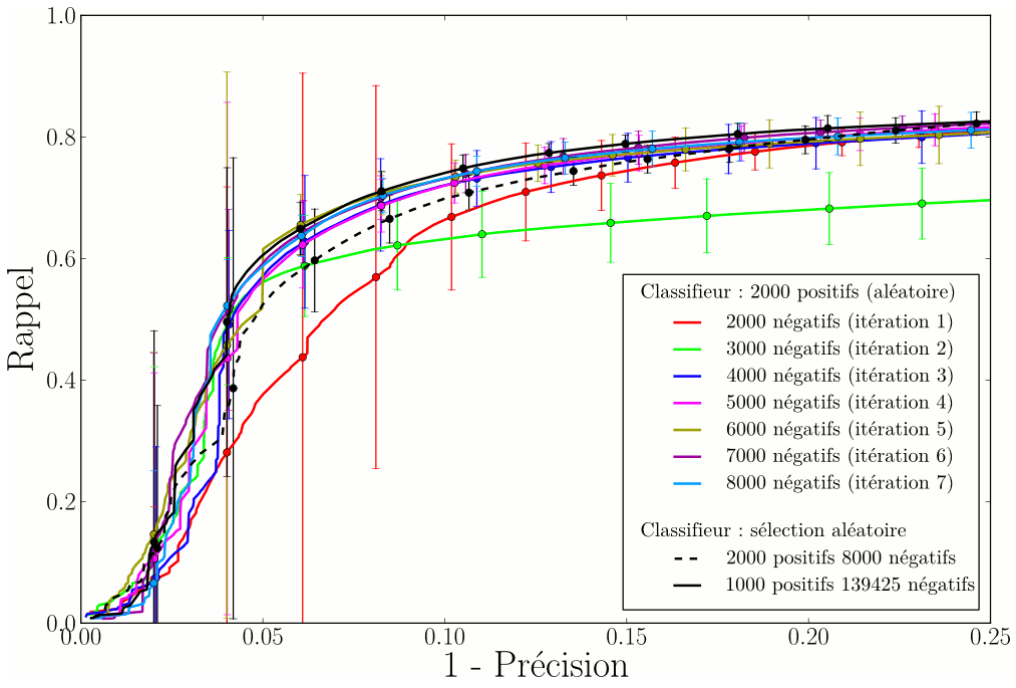


FIGURE 5.5 – Moyennes et variances des cinq classifieurs des courbes précision-rappel pour les six itérations du *bootstrapping*.

TABLE 5.1 – Caractéristiques moyennes des classifieurs après chaque itération de *bootstrapping*.

	Rappel, (écart type)	Précision	F-Mesure
2000 positifs, 2000 négatifs	0,77 (0,018)	0,83	0,80
2000 positifs, 3000 négatifs	0,65 (0,034)	0,87	0,75
2000 positifs, 4000 négatifs	0,75 (0,021)	0,87	0,81
2000 positifs, 5000 négatifs	0,76 (0,015)	0,86	0,81
2000 positifs, 6000 négatifs	0,75 (0,015)	0,88	0,81
2000 positifs, 7000 négatifs	0,77 (0,014)	0,86	0,81
2000 positifs, 8000 négatifs	0,76 (0,013)	0,87	0,81
2000 positifs, 8000 négatifs (aléatoire)	0,76 (0,011)	0,85	0,80
1000 positifs, 139425 négatifs (aléatoire)	0,77 (0,008)	0,87	0,82

Les résultats montrent, pour une précision élevée (plus de 85%), que le *bootstrapping* permet avec moins d'exemples négatifs de se rapprocher fortement des meilleurs classifieurs obtenus sur cette séquence. En dessous de 85% de précision, toutes les courbes se rejoignent. Cette tendance est confirmée par la table 5.1. Elle indique les performances optimales au sens de la F-Mesure des différents classifieurs et montre que tous les points, avec ou sans *boostrapping*, sont relativement proches. De plus la variance dans les résultats reste du même ordre de grandeur dans les deux cas. Elle diminue cependant au fur et à mesure des itérations pour se rapprocher de celle obtenue avec une sélection purement aléatoire.

Enfin, les performances des classifieurs obtenus après seulement une itération sont nettement inférieures aux autres. Le *bootstrapping* lors de sa première itération augmente l'importance des exemples les plus difficiles. Dans le cas de la contextualisation automatique, cette approche peut poser quelques problèmes. En effet en se concentrant sur les exemples difficiles, il est possible d'augmenter le poids des exemples mal labellisés et donc de perturber l'apprentissage. L'apport de nouveaux exemples dans les itérations suivantes peut diluer ces exemples mal labellisés, ce qui explique la hausse des performances.

Ces résultats montrent que le *bootstrapping* peut être une technique plus intéressante que la sélection purement aléatoire et cela même lorsque la base est légèrement bruitée. Il permet d'augmenter le rappel dans le cas d'une précision élevée, ce qui est le but de toute amélioration des performances.

5.3.2 Exemples positifs

Après s'être intéressé à la sélection des exemples négatifs et montré que le *bootstrapping* était une réponse efficace à ce problème, nous nous proposons d'expérimenter deux stratégies pour sélectionner les exemples positifs. Ces dernières ont pour but de construire une base la plus diversifiée possible, tout en diminuant le nombre d'exemples nécessaires ainsi que la variance dans les performances des classifieurs.

5.3.2.1 Sélection par apparence : clustering

Le but de la sélection des exemples positifs est de choisir les 2 000 exemples les plus divers parmi ceux étiquetés par l'oracle. Une idée pour parvenir à ce résultat est de regrouper les exemples les plus similaires de la base entre eux et de prendre l'exemple le plus significatif de chaque cluster.

Le k-means (ou k-moyennes en français) est un algorithme non supervisé dont le but est de regrouper les éléments d'une base qui sont proches. Contrairement à l'algorithme du mean shift, le k-means a besoin pour fonctionner de connaître le nombre de clusters souhaité. [Arthur et Vassilvitskii, 2007] a récemment proposé une stratégie pour initialiser de manière optimale le k-means. C'est donc cette variante que nous utiliserons par la suite. La procédure de l'algorithme k-means++ est détaillée sur la figure 7.

La sélection des exemples consiste, dans cette expérience, à effectuer un k-means sur la base des exemples positifs. L'exemple le plus proche de la moyenne de chaque cluster est ensuite intégré à la base d'apprentissage. La distance utilisée dans cette expérience est la norme euclidienne. Le paramètre k indique donc le nombre d'exemples positifs souhaité lors d'un apprentissage.

L'expérience a été réalisée cinq fois de suite sur la séquence ViCoMo 2, sans *bootstrapping* afin de n'étudier que la sélection des exemples positifs. La sélection des exemples positifs a été testée avec deux ratios différents d'exemples positifs et d'exemples négatifs : 200 positifs et 4 000 négatifs d'une part et 2 000 positifs et 8 000 négatifs d'autre part.

Les résultats sont présentés sur la figure 5.6. Les classifieurs entraînés avec une base échantillonnée à l'aide du k-means sont comparés à d'autres entraînés avec une base échantillonnée aléatoirement.

Algorithme 7: Algorithme du k-means avec initialisation du type k-means++

Entrées :
Un ensemble de points $X = (x_i)_{1 \leq i \leq n} \in \mathbb{X}$
Le nombre désiré k de clusters, numérotés de 1 à k
Une distance D dans l'espace des descripteurs (distance euclidienne)
Le nombre d'itérations maximales $maxIt$ (10 par défaut)

Sortie :
L'ensemble L contenant les affectations $l_i \in \{1..k\}$ de chaque descripteur x_i

Algorithme :

```
// Initialisation (k-means++)
```
Choisir aléatoirement un centre c_0 parmi tous les points de X
Placer c_0 dans l'ensemble C des centres des clusters

tant que $card(C) < k$ **faire**
  Initialiser chaque $(d_i)_{1 \leq i \leq n}$ à l'infini
  d_i est la distance minimale d'un point à un centre
  pour chaque $x_i \in X$ **faire**
    pour chaque $c_j \in C$ **faire**
      $d_i = min(d_i, D(x_i, c_j))$
    fin
  fin
  Tirer un point x_i avec une probabilité proportionnelle à d_i^2 Ajouter le nouveau centre x_i dans C
fin

```
// Début de la procédure du k-means
```
$nbIt = 0$
tant que $nbIt < maxIt$ **faire**
  Incrémenter $nbIt$
  ```
  // Affectations des descripteurs à un cluster
  ```
  Initialiser chaque $(d_i)_{1 \leq i \leq n}$ à l'infini
  pour chaque $x_i \in X$ **faire**
    pour chaque $c_j \in C$ **faire**
      Calculer la distance $d = D(x_i, c_j)$.
      si $d < d_i$ **alors**
        $d_i = d$ et $l_i = j$
      fin
    fin
  fin
  ```
  // Mise à jour des centres
  ```
  pour chaque $l_i \in L$ **faire**
    Calculer la moyenne des descripteurs du cluster l_i
    Remplacer c_{l_i} par cette moyenne.
  fin
fin

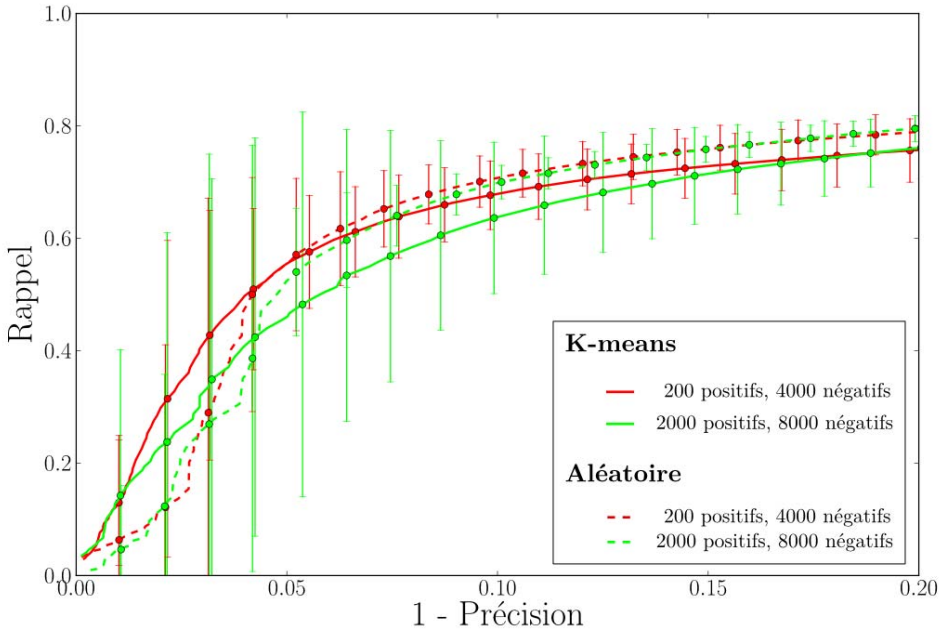


FIGURE 5.6 – Courbes moyennes de la précision en fonction du rappel de classifieurs entraînés avec une base de positifs échantillonnée à l'aide du k-means ou avec une base échantillonnée aléatoirement.

Les courbes montrent que pour les précisions très élevées (supérieure à 95%), la sélection des exemples positifs peut améliorer les performances. Mais le rappel dans cette zone de fonctionnement des classifieurs est trop faible pour que le classifieur y soit exploitable en pratique. Pour les précisions plus faibles, la sélection des exemples par k-means diminue le rappel par rapport à une sélection aléatoire. Enfin la variance des résultats n'est en aucun cas réduite par cette approche.

La sélection des exemples positifs à l'aide du k-means ne semble donc pas une bonne approche pour la contextualisation. Le prochain paragraphe tente une nouvelle approche pour la sélection à l'aide d'un tracker.

5.3.2.2 Sélection par suivi

L'objectif de l'expérience précédente est de sélectionner moins d'exemples positifs, tout en maintenant dans la base la plus grande diversité possible. Cette expérience n'est pas très concluante, mais il est difficile de savoir si cela provient de sa mise en œuvre ou de l'idée même de la sélection des exemples positifs.

Pour tenter d'apporter une réponse à cette question, une nouvelle expérience a été réalisée. Son but est une nouvelle fois de sélectionner les exemples positifs de manière à maximiser leur diversité dans la base.

En effet, lors de l'étude de la base d'apprentissage, la première chose qui frappe est que de nombreux exemples sont en fait des observations de la même personne pris à

des intervalles de temps très courts. Ces exemples pris dans leur ensemble ne semblent pas apporter beaucoup plus d'informations qu'un seul pris parmi eux. Réaliser un apprentissage avec ces exemples peut conduire à une dégradation des performances à cause du manque de diversité. Le tracking devrait permettre de supprimer efficacement les multiples apparitions d'une même personne et d'augmenter la diversité des exemples dans la base. Contrairement à ce qui a été étudié dans le cadre de l'oracle dans le paragraphe 4.3.5.1 (page 87), le but n'est pas ici d'augmenter le rappel en ajoutant des exemples dans la base. L'idée est de ne sélectionner qu'un petit nombre d'exemples par piste. La supposition sous-jacente est que deux exemples issus d'une même piste sont plus proches, se ressemblent plus que deux exemples provenant de pistes distinctes.

D'un point de vu pratique, cette expérience nécessite un tracker performant capable de suivre de nombreuses personnes pendant un intervalle de temps assez long. Ce n'est pas forcément le cas de celui utilisé conjointement avec l'oracle. Afin d'éviter les aléas liés au tracking, l'expérience a été réalisée avec un tracker parfait dont les pistes ont été annotées par un humain.

La sélection des exemples est alors effectuée comme suit : pour chaque piste, un nombre limité mais constant d'exemples est choisi de manière aléatoire. Il est ainsi impossible pour un piéton d'occuper toute la base, même s'il est resté très longtemps dans la scène.

Afin d'évaluer cette approche, nous comparons les classifieurs obtenus avec d'autres, entraînés avec le même nombre d'exemples mais sélectionnés aléatoirement.

L'expérience a été réalisée sur la séquence ViCoMo 1, car c'est elle qui possède la plus longue annotation réalisée par un humain. Les 10 000 premières images permettent de construire la base tandis que les images numérotées de 15 000 à 16 000 servent de vérité terrain. Sur les 10 000 premières images, 181 pistes ont été extraites. Les résultats sont exposés sur les figures 5.7 et 5.8. Elles correspondent aux résultats moyens obtenus pour des classifieurs entraînés avec 5, 8, 10, 13 ou 15 exemples par pistes, ce qui correspond respectivement à 905, 1 448, 1 810, 2 353 et 2 715 exemples au total, l'idée étant de couvrir la plage entre 1 000 et 2 000 exemples positifs.

L'étude de ces courbes indique que la sélection à l'aide du tracking n'apporte aucun avantage. Pire, pour les précisions élevées, le rappel diminue dans certains cas. Le tracking n'apporte donc aucun avantage pour la sélection des exemples positifs.

Au travers de ces deux expériences (k-means et tracking), il n'a pas été possible de trouver une sélection des exemples positifs pertinente qui apporte un avantage lors de l'apprentissage. Cela semble confirmer que le choix des exemples positifs jouent un rôle moindre que celui des exemples négatifs lors de l'apprentissage d'un classifieur contextualisé.

La partie suivante tente de revoir l'approche de la sélection des exemples en s'intéressant non plus uniquement aux exemples mais aussi à leur contexte spatial.

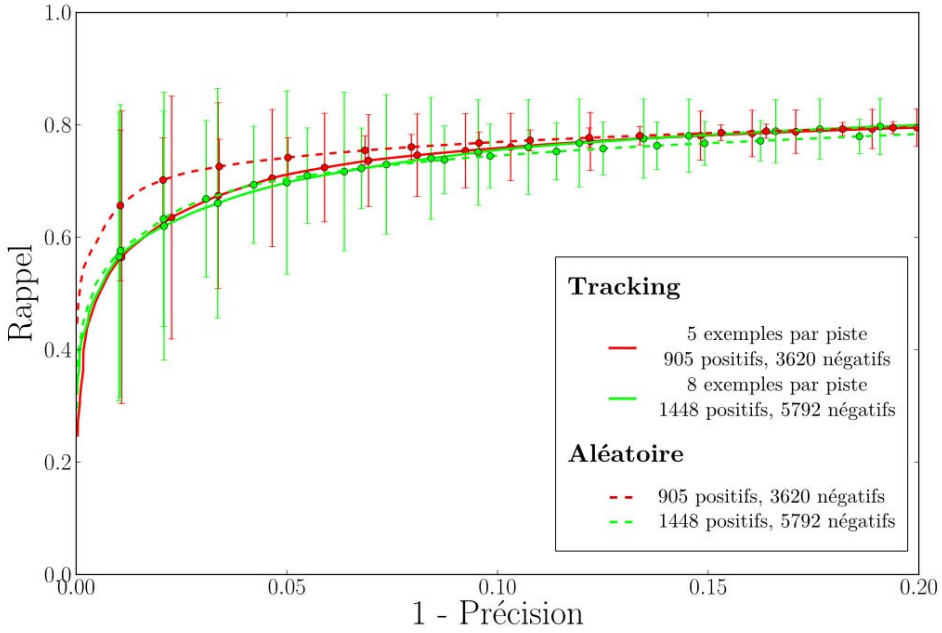


FIGURE 5.7 – Comparaison d'un apprentissage avec sélection aléatoire ou avec sélection à l'aide du tracking avec 5 ou 8 exemples par pistes.

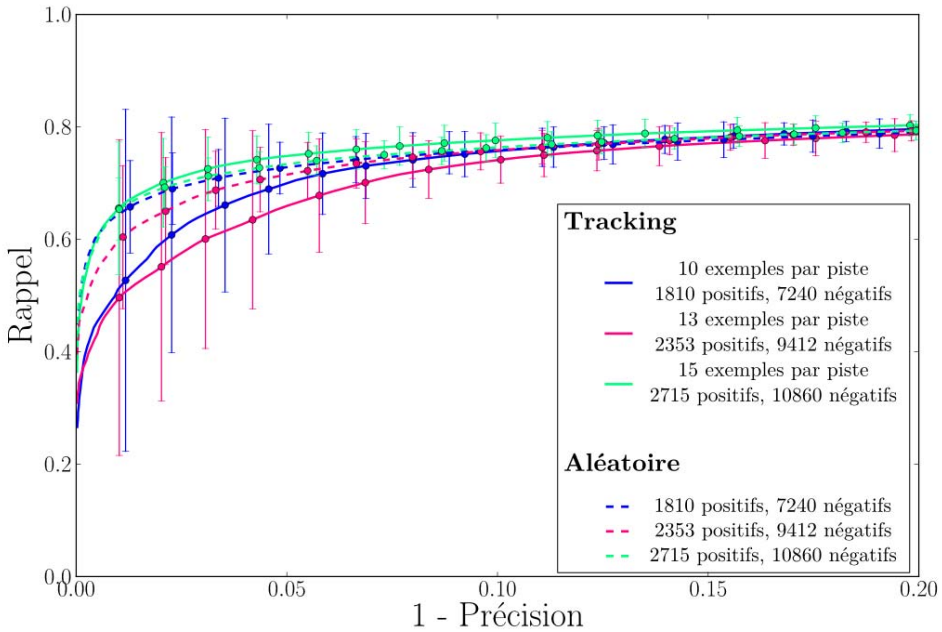


FIGURE 5.8 – Comparaison d'un apprentissage avec sélection aléatoire ou avec sélection à l'aide du tracking avec 10, 13 ou 15 exemples par pistes.

5.4 Sélection par contraintes géométriques et spatiales

À l'issue de la partie sur la sélection aléatoire des exemples, nous avons constaté une saturation des performances. De nombreux classifieurs, malgré des conditions d'entraînement différentes, offrent des résultats similaires et ne dépassent pas 70% de rappel pour 80% de précision.

Nous avons supposé que ce phénomène pouvait peut-être s'expliquer grâce aux suppositions suivantes. D'une part, la stratégie de sélection des exemples ne serait pas adéquate et le classifieur ne serait pas entraîné sur les bons exemples présents dans la base. D'autre part, le modèle de représentation des piétons serait trop peu expressif et ne permettrait pas de prendre en compte toute la diversité des exemples présents dans la scène.

Les expériences menées précédemment ont démontré qu'une bonne stratégie de sélection des exemples, comme le *boostrapping* ont une influence sur les performances. Néanmoins, il n'a pas été possible de dépasser les performances des meilleurs classifieurs entraînés avec des exemples choisis au hasard. Dans cette partie nous tentons d'explorer une autre possibilité à l'aide d'une sélection des exemples qui prend en compte les contraintes géométriques et spatiales qui s'exercent sur la scène.

5.4.1 Intérêts d'un découpage par régions

L'apparence des piétons peut varier très fortement suivant leur position dans l'image. Par exemple pour une caméra de vidéosurveillance qui est généralement placée en hauteur, il n'est pas rare que les piétons situés en bas de l'image soient grands et vus en plongée et que les piétons situés en haut de l'image soient petits et vus de face. À cela s'ajoutent les déformations qui apparaissent à droite et à gauche de l'image qui ont tendance à pencher les piétons. Toutes ces distorsions perturbent le détecteur car elles invalident le modèle générique appris (piéton vertical, de taille moyenne et vu de face). D'autre part en apprentissage, il est plus facile de discriminer deux classes si elles sont bien séparées et compactes dans l'espace des descripteurs. Or un changement d'apparence dans l'image peut conduire à des modifications importantes du descripteur, ce qui explique pourquoi la gestion des différentes échelles dans un détecteur est problématique [Benenson *et al.*, 2012].

Plusieurs stratégies ont été élaborées pour simplifier et restreindre le problème de l'apprentissage. Par exemple [Wu et Nevatia, 2007a] architecturent leur classifieur sous forme d'arbre et regroupent les descripteurs qui se ressemblent pour entraîner chaque nœud.

D'autre part, pour réussir à détecter des piétons dans des postures différentes, il existe des classifieurs basés sur des modèles par parties comme les *Deformable Parts Models* (ou DPM) proposés par [Felzenszwalb *et al.*, 2008]. Au moment de la détection, chaque partie peut bouger indépendamment de ses voisines. Le score global de classification prend en compte le score de chaque partie ainsi que sa localisation par rapport aux autres. L'intérêt de cette approche est sa plus grande souplesse. Le modèle appris est très générique et un piéton légèrement penché peut être détecté grâce au fait que le modèle est capable de prendre en compte les déformations.

Dans notre cas, ces approches ne sont pas optimales. Le DPM en particulier génère un surcoût important au moment de l'apprentissage (ce qui n'est pas trop pénalisant) mais surtout à la détection ce qui n'est pas souhaitable. Nous connaissons déjà la scène

sur laquelle va être utilisé le détecteur. Elle est fixe et présente une structure qui influe fortement sur l'apparence et la densité des piétons. En analysant la géométrie de la scène et la perspective, il est possible de simplifier le modèle utilisé en un point donné. La complexité du modèle est externalisée au niveau du détecteur qui doit alors gérer plusieurs modèles. Cette approche évite notamment des calculs importants lors de la détection puisqu'un seul modèle, rigide et donc plus simple, est testé à un endroit donné de la scène.

Les *classifier grids* [Grabner *et al.*, 2007] (*cf* paragraphe 2.3.3.3, page 36) exploitent cette idée et vont jusqu'à entraîner un classifieur pour chaque point dans l'image. Mais ces méthodes très locales peuvent être très sensibles au bruit. Par exemple, une erreur d'étiquetage d'une observation en un point donné ne peut pas être atténuée par les bonnes labellisations des observations du voisinage et va donc dégrader les performances localement.

Une solution intermédiaire adaptée au contexte routier a été proposée par [Park *et al.*, 2010]. Elle consiste à utiliser deux modèles de piétons afin de tenir compte de la perspective. Le premier se concentre sur les piétons du premier plan qui par définition possèdent une résolution importante et pour lesquels il est possible d'observer des postures différentes. Pour prendre en compte toute cette information, les auteurs ont fait le choix d'un modèle par parties [Felzenszwalb *et al.*, 2008]. Le second modèle est adapté aux piétons éloignés et donc plus petits dans l'image. Comme ils possèdent moins de détails, un modèle plus simple est utilisé.

Dans notre cadre applicatif, la caméra est fixe. Les différentes apparences des piétons dans l'image sont donc stables dans le temps. Les déterminer et définir les régions dans lesquelles elles apparaissent peut permettre d'intégrer une connaissance locale de la scène dans le détecteur et ainsi de répondre de manière spécifique aux problèmes rencontrés. Avoir un modèle unique de représentation des piétons pour toute la scène n'est pas adapté, puisque cela nécessite qu'il capte toutes les apparences possibles et qu'en conséquence il soit plus complexe.

Nous avons donc choisi de créer plusieurs modèles, un par région d'intérêt. L'avantage est que chaque modèle est dédié localement. L'apprentissage du classifieur est plus simple car la variabilité des exemples à l'intérieur d'une classe est réduite. D'autre part, le modèle n'est pas plus complexe localement et aucun calcul supplémentaire n'est requis par rapport à une version avec un modèle global qui parcourrait toute l'image. Au final la représentation est plus riche car mieux adaptée localement avec un surcoût présent uniquement à l'apprentissage.

Une fois les régions définies, elles peuvent se révéler utiles lors des deux phases principales. Lors de l'apprentissage, comme dit précédemment, il paraît important de dissocier les apparences différentes entre les piétons et ainsi obtenir une meilleure séparation des exemples.

Mais cela peut aussi être un atout lors de la détection. En effet, un classifieur global peut répondre plus ou moins bien en fonction de la partie de l'image qu'il explore. Sans un découpage de l'image en régions, il est difficile de régler le détecteur différemment en fonction des difficultés qu'il rencontre. Cela n'est bien sûr pas le cas si des régions ont été définies. Chaque classifieur peut, si besoin, être ajusté localement de manière totalement indépendante. Nous reviendrons sur le réglage du détecteur dans le chapitre 6.4.

5.4.2 Création des régions

Les régions permettent de renforcer la localité dans l'image des exemples appris par un classifieur. Cette localité favorise l'apprentissage de piétons qui ont une taille et une inclinaison similaires à cause de la perspective. Il est donc important que les régions soient les plus compactes possible et ne s'étendent pas sur de grandes portions de l'image.

D'autre part dans une image, certaines structures peuvent localement perturber la réponse du classifieur. Par exemple une structure verticale comme un poteau peut ressembler suffisamment à un piéton pour avoir un score élevé. Mettre tous les exemples le contenant dans la même zone peut permettre de limiter son influence à cette région et laisser les autres intactes. Pour cette raison il peut être judicieux de placer cette structure dans une seule zone. Néanmoins si la structure occupe une large place dans l'image, il ne faut pas qu'elle conduise à la création de régions dans lesquelles les piétons ne se ressemblent pas.

Pour fabriquer les régions, nous avons décidé de travailler directement avec les détections de l'oracle. L'idée est de les regrouper à l'aide d'un algorithme non-supervisé. Pour que les régions construites soient compactes et que les structures complexes n'en perturbent pas plusieurs, nous prenons en compte la localisation des exemples dans l'image (ou dans le plan du sol en 3D) et le score de classification du détecteur.

5.4.2.1 Superpixel

Le partitionnement d'une image en superpixels, c'est-à-dire en clusters plus ou moins réguliers de pixels, est une étape primordiale dans de nombreux systèmes. Les algorithmes qui découpent une image en superpixels se basent sur des informations de couleur et de position des pixels pour définir les frontières. Idéalement, une région doit contenir des pixels homogènes tandis que deux clusters différents ne doivent pas être similaires.

De nombreuses méthodes ont été proposées dans la littérature parmi lesquelles [Ren et Malik, 2003; Felzenszwalb et Huttenlocher, 2004]. Ils utilisent des algorithmes non supervisés pour regrouper les pixels similaires. Nous nous sommes plus particulièrement intéressés au SLIC [Achanta *et al.*, 2012] (pour *simple linear iterative clustering*), car il est relativement simple à mettre en œuvre et donne de bons résultats. Bien que pour notre application cela ne soit pas un critère important puisque les zones ne sont construites qu'une seule fois lors de l'apprentissage contextualisé, cette méthode est parmi les plus rapides. La figure 5.9 présente un exemple de segmentation et l'algorithme 8 présente la procédure des SLIC.

L'image est d'abord convertie de l'espace couleur RGB vers une représentation CIELAB. Cet espace de représentations des couleurs a été choisi car il a été spécialement mis au point pour que les distances entre couleurs soient proches de celle perçues par l'oeil humain. L'algorithme travaille dans un espace à cinq dimensions : la position (x, y) du pixel et les coordonnées des couleurs (l, a, b). SLIC est un algorithme itératif similaire à un k-means (*cf* algorithme 7) légèrement modifié. Les centres des clusters $c_i = \{x_i, y_i, l_i, a_i, b_i\}$ sont initialisés selon une grille régulière dans toute l'image. Le pas de la grille est de $S = \sqrt{\frac{N}{k}}$ pixels et dépend du nombre k de superpixels désiré et du nombre de pixels N dans l'image. Intervient ensuite le processus itératif divisé en deux étapes.

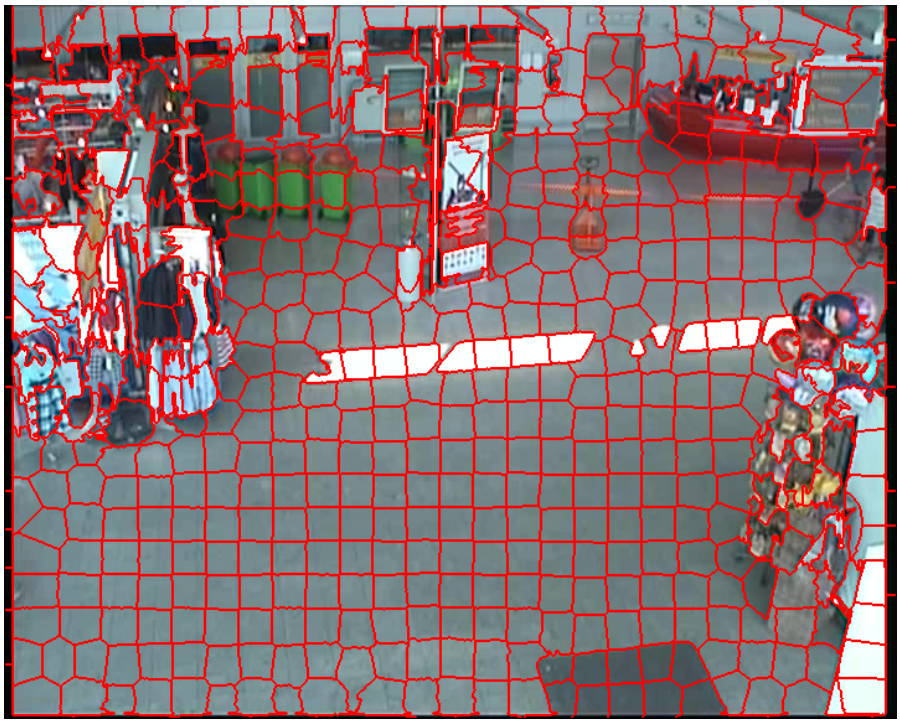


FIGURE 5.9 – Exemple de segmentation en superpixels d'une image de ViCoMo 1. Les pixels qui partagent localement des propriétés de couleur proches sont regroupés. Les frontières des clusters sont indiquées en rouge. La finesse des clusters dépend du nombre de superpixels souhaité.

La première assigne un cluster à chaque pixel : un pixel est associé au centre le plus proche dans un rayon de $2S$. Contrairement au k-means classique qui rechercherait les associations dans tout l'espace (et donc dans toute l'image), le SLIC se restreint spatialement. Cela permet d'assurer une plus grande compacité aux superpixels mais aussi d'accélérer le temps de traitement. L'espace de travail, qui est la concaténation entre l'espace de position et celui de couleur, a cinq dimensions. La distance utilisée doit donc y être adaptée. Les auteurs ont choisi la distance euclidienne dans ces deux espaces avec un coefficient α de normalisation pour pouvoir les comparer. Le coefficient α est constant et ajusté empiriquement. Si la couleur a une influence trop faible, les superpixels ne correspondent pas aux frontières dans l'image. Dans cet espace, la distance entre deux éléments p_i et p_j est la suivante :

$$D(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + \alpha[(l_i - l_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2]}$$

La deuxième étape met à jour la position des centres en calculant la moyenne de tous les vecteurs des pixels qui appartiennent à ce cluster. Une nouvelle itération est lancée jusqu'à ce que la limite d'itérations soit atteinte. En pratique cette limite est fixée à 10.

Enfin, une fois le nombre d'itérations maximal atteint, une étape de post-traitement est réalisée pour agréger les clusters trop petits avec un voisin plus grand.

Algorithme 8: Algorithme des superpixels : SLIC

Entrées :
Une image RGB
Le nombre désiré k de superpixels, numérotés de 1 à k
Une distance D dans l'espace (x, y, l, a, b)
Le nombre d'itérations maximal $maxIt$ (10 par défaut)

Sortie :
L'ensemble L contenant les affectations $l_i \in \{1..k\}$ de chaque pixel p_i

Algorithme :

Initialiser les centres des clusters c_i selon une grille régulière de pas S.
Initialiser l'affectation l_i de chaque pixel à -1
Initialiser la distance du pixel au centre du cluster associé d_i à $+\infty$
Convertir l'image vers la représentation CIELAB.

```
// Début de la procédure du k-means
```
$nbIt = 0$
tant que $nbIt < maxIt$ **faire**
  Incrémenter $nbIt$
  ```
  // Affectations des pixels à un cluster
  ```
  pour chaque *centre* c_i **faire**
    pour chaque *pixel* p_j *dans une zone de taille* $2S \times 2S$ *autour de* c_i **faire**
      Calculer la distance $d = D(c_i, p_j)$ entre c_i et p_j.
      si $d < d_j$ **alors**
        $d_j = d$
        $l_j =$ i
      fin
    fin
  fin
  ```
  // Mise à jour des centres
  ```
  pour chaque $l_i \in L$ **faire**
    Calculer la moyenne des vecteurs des pixels ayant le label l_i.
    Remplacer c_i par cette valeur.
  fin
fin

5.4.2.2 Application à la création des régions

Nous disposons de la calibration de la caméra et l'oracle 3D a collecté un ensemble d'exemples positifs. Nous avons décidé de nous inspirer de l'algorithme SLIC pour définir de manière automatique les différentes régions. Contrairement à ce dernier, nous allons définir nos régions dans le plan du sol et non dans le plan de l'image. Nous allons donc travailler directement avec les positions des détections 3D fournies par l'oracle et non avec les données des pixels.

D'autre part, nous avons besoin d'un critère pour décider si deux piétons se res-

semblent. Une possibilité est de travailler sur le descripteur, mais ce vecteur est dans un espace de grande dimension ce qui complique fortement les traitements possibles. Notre choix s'est porté sur l'utilisation du score fourni par le classifieur générique. Deux scores sont extraits, pour chaque position 3D où un piéton a été détecté par l'oracle à un instant t quelconque. Le score dit « score des positifs » s_p correspond simplement à la sortie du classifieur à l'instant t. Le score dit « score des négatifs » s_n est constitué par la moyenne des scores du classifieur à cet endroit, à tous les instants sauf t. À la manière de la soustraction de fond, cette moyenne étant effectuée sur un temps assez long, la présence d'un piéton pendant quelques images n'a pas d'incidence particulière. Ces deux scores indiquent la difficulté du classifieur à détecter les piétons à cet endroit. Par exemple, un score des négatifs élevé signifie un risque d'avoir des faux positifs. Avec un score des positifs faible il est probable que tous les piétons ne soient pas détectés car ils ressemblent pour le classifieur à du fond.

Comme les régions sont construites pour le détecteur contextualisé et non pour le détecteur générique, il semblerait plus intéressant de travailler avec les scores provenant d'un classifieur entraîné à partir des exemples de la scène. Cependant à ce stade de la contextualisation nous ne disposons pas d'un tel classifieur. Pour limiter le nombre d'apprentissages déjà important avec le découpage en régions, nous avons préféré utiliser le classifieur dont nous disposions, c'est-à-dire le générique. De plus, il est possible que le classifieur contextualisé ayant déjà appris l'apparence des piétons dans la scène crée des scores plus uniformes spatialement, rendant le travail de constitution des zones plus délicat.

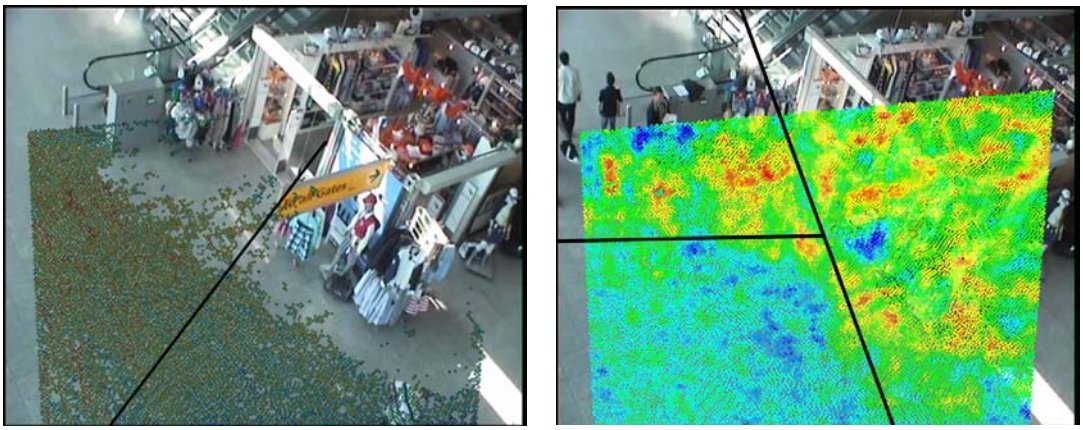
Nous nous retrouvons donc dans le même cas que l'algorithme SLIC à vouloir fusionner deux types d'informations : spatiale et d'apparence. Nous définissons un espace à quatre dimensions (x, y) les coordonnées de l'exemple dans le plan du sol et (s_p, s_n) les scores du classifieur en ce point.

L'intérêt des régions réside dans le fait de séparer les apparences des piétons pour les affecter à des classifieurs différents. Pour vérifier leur potentiel, nous les avons testées sur la scène ViCoMo 2. En effet les piétons y présentent de fortes variations dues à la perspective. Dans le fond de l'image, ils sont droits et vus de trois-quart ou de face, de manière similaire à PETS 2006. En bas vers la gauche, ils sont vus de haut avec une vue en plongée. Puis plus les piétons sont situés à droite plus ils sont penchés dans l'image. C'est en bas à droite, que la perspective est la plus forte. Les piétons y sont fortement penchés et les boîtes du détecteur en 3D ne respectent plus le ratio 0,5 entre la largeur et la hauteur de la boîte.

Instinctivement, il faudrait donc diviser la scène en trois ou quatre régions. La figure 5.10 présente un découpage avec quatre régions fabriqué à la main qui peut sembler naturel pour cette scène. Les piétons du fond étant plus éloignés de la caméra, leur apparence est plus stable sur une plus grande portion de l'image. À l'inverse, dans le bas de l'image, deux ou trois régions sont nécessaires. La zone violette est une transition entre la zone verte contenant des piétons bien droits et la bleue où les piétons sont fortement penchés.

Le point de départ de l'expérience consiste à extraire, grâce à l'oracle 3D, les positions dans le plan du sol des exemples positifs (*cf* figure 5.11(a)) et une carte du score des négatifs (*cf* figure 5.11(b)).

FIGURE 5.10 – Découpage de la scène réalisé par un humain afin de tenir compte de l'apparence des piétons.



(a) Carte de score des exemples positifs (b) Carte de score des exemples négatifs

FIGURE 5.11 – Pour chaque piéton extrait par l'oracle, la figure (a) montre la carte des scores des positifs s_p et la figure (b) présente la carte des scores des négatifs s_n. Sur les deux cartes, le bleu indique des valeurs faibles et le rouge des valeurs élevées. Les traits noirs correspondent à des frontières qu'un humain pourrait tracer à partir de ces données.

Maintenant, nous allons appliquer le k-means modifié des superpixels aux données ainsi rassemblées. À chaque exemple est assignée une région (figure 5.12). Pour cette scène nous avons demandé à l'algorithme de fabriquer quatre régions.

Il est peu probable que l'oracle ait labellisé au moins un piéton pour chaque point du plan du sol. À ce stade, toutes les hypothèses, qui vont être testées par le détecteur, n'ont donc pas été affectées à une région. Un algorithme des plus proches voisins permet d'affecter à chaque point restant la région à laquelle il appartient. Cette région est simplement celle de l'exemple, issu de l'oracle, le plus proche de ce point.

La figure 5.13 montre le découpage final des régions et l'aspect des exemples dans celles-ci. Dans chaque région, les piétons partagent une apparence similaire ce qui n'est pas forcément le cas entre deux régions différentes.

FIGURE 5.12 – Découpage de la scène réalisé par l'algorithme des superpixels.

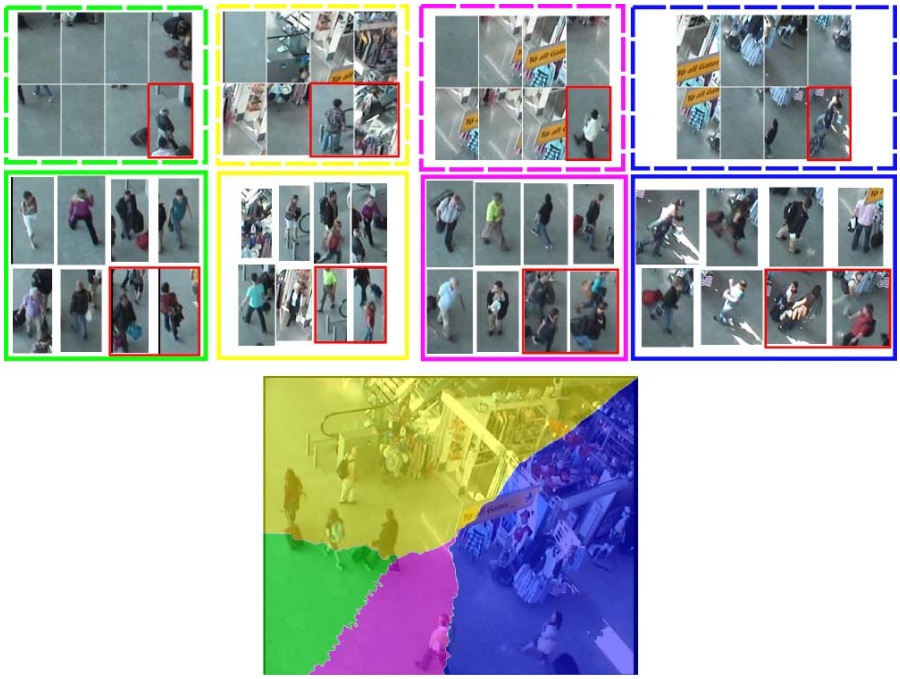


FIGURE 5.13 – Apparence des exemples pour chaque région. Les exemples entourés par des pointillés correspondent aux exemples négatifs, ceux entourés en traits pleins correspondent aux exemples positifs. Les erreurs de labellisation sont indiquées en rouge.

Le coût de construction des régions est assez faible. En effet une fois que l'oracle a fabriqué la base et même si elle contient beaucoup de piétons, les vecteurs utilisés ne sont que de dimension quatre. L'algorithme est donc très rapide (environ 160 millisecondes pour cette scène avec 20 000 exemples) et peut donc être relancé régulièrement si le besoin s'en fait sentir. Le seul véritable coût lors de l'utilisation des zones est la nécessité de calculer plusieurs classifieurs, là où un seul suffisait précédemment.

5.4.3 Validations expérimentales

5.4.3.1 Résultats par région

Les régions sont désormais créées. Il reste maintenant à vérifier que leur utilisation a un impact bénéfique sur les performances de notre détecteur contextualisé. Dans ce paragraphe sont étudiées les performances de chaque classifieur dans leur région. Le nombre d'exemples entraîné pour construire un classifieur est le même dans toutes les régions et ne tient pas compte de la surface de la région ou de la taille de la base collectée par l'oracle. 2 000 exemples positifs et 8 000 exemples négatifs sont sélectionnés aléatoirement dans chaque région.

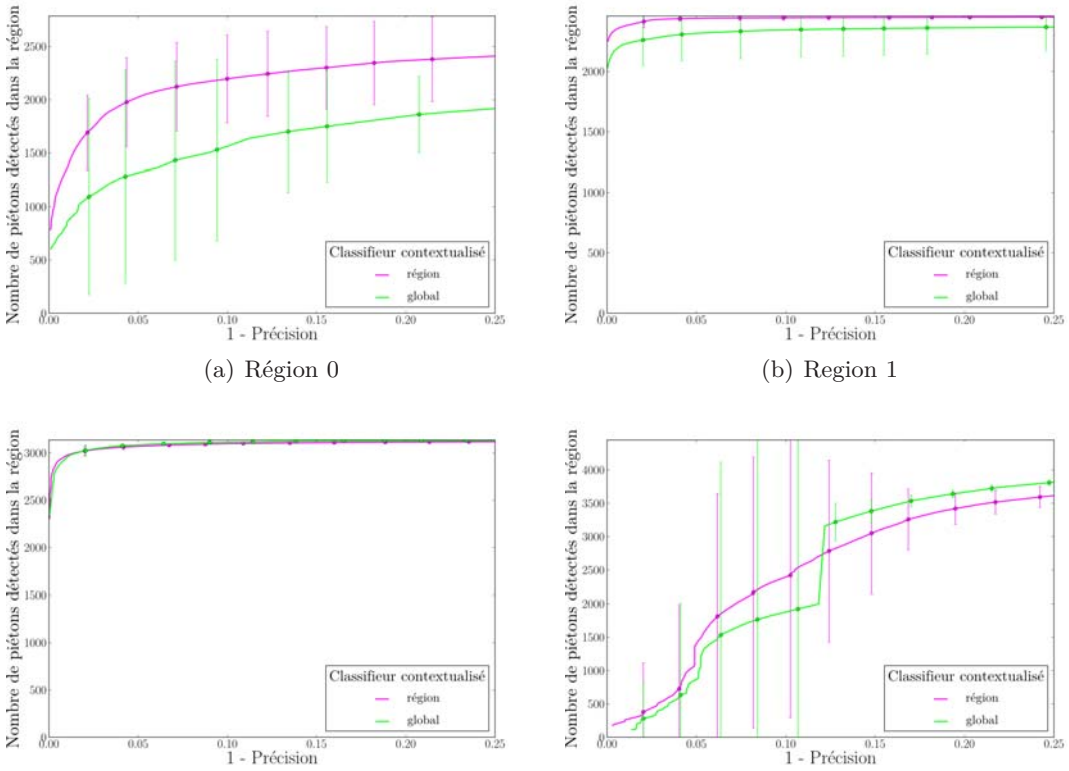
Pour montrer l'intérêt d'un découpage de la scène, chaque classifieur régional est comparé avec un autre qui a été entraîné de manière globale. Les exemples de ce dernier sont aussi sélectionnés de manière aléatoire dans l'image et l'ensemble d'apprentissage comporte également 2 000 exemples positifs et 8 000 exemples négatifs.

Le protocole expérimental décrit dans le chapitre présentant la contextualisation (*cf* paragraphe 3.3.1, page 45) est à nouveau utilisé. Dans le cas des classifieurs entraînés par région, il peut y avoir des problèmes car il est possible qu'une personne soit à cheval sur deux d'entre elles. Les deux classifieurs concernés vont logiquement la détecter mais comme le piéton n'appartient qu'à une zone, les détections d'un classifieur risquent d'être mal notées. Pour corriger cela nous avons décidé de définir deux critères supplémentaires, un pour le rappel et un pour la précision.

- rappel : un piéton est compté positivement uniquement si ses pieds (le point situé au centre et en bas de la boîte de détection) sont dans la zone considérée ;
- précision : une détection est considérée comme étant un faux positif uniquement si elle n'est pas similaire à une annotation de la vérité terrain, c'est-à-dire que le centre de la détection n'est pas contenu dans l'ellipse d'une annotation, même si cette dernière n'est pas dans la région étudiée.

Les résultats pour chaque région sont présentés sur la figure 5.14. Nous constatons une augmentation significative des performances dans la région 0. Cette région est située en bas à droite et contient les piétons les plus penchés. Dans la région 1, l'amélioration est nettement plus légère. Là encore il s'agit d'une région qui comporte des piétons dont l'apparence est relativement rare dans la scène. Il n'y a aucun changement significatif dans les régions 2 et 3. La forme particulière des courbes dans la région 3 provient du fait que cette portion de l'image contient des exemples difficiles. L'explication est la même que celle donnée dans le chapitre sur les oracles. L'escalier mécanique dans le fond couplé aux vêtements a tendance à provoquer des mauvaises détections pour l'oracle ce qui vient perturber l'apprentissage dans cette région par la suite. L'intérêt des régions dans ce cas est de ne pas polluer les classifieurs voisins avec de mauvais exemples en compartimentant les erreurs.

Ceci semble démontrer que le découpage en régions permet d'améliorer les performances du détecteur contextualisé. Dans deux des quatre régions, les classifieurs sont meilleurs que le classifieur global. Dans les deux autres régions, les performances se maintiennent au même niveau. Le prochain paragraphe est consacré à l'étude des performances globales du système.



(a) Région 0 (b) Region 1

(c) Région 2 (d) Region 3

FIGURE 5.14 – Courbes indiquant le nombre de piétons détectés dans la région en fonction de la précision du classifieur.

5.4.3.2 Résultats globaux

Le paragraphe précédent a permis l'étude des performances de chaque classifieur séparément. Cependant le plus intéressant pour la détection est de savoir si le détecteur par région est meilleur que le détecteur entraîné sur toute l'image. Une méthode pour obtenir une courbe globale pour les deux détecteurs est donc nécessaire.

Pour le classifieur qui a été entraîné sur toute l'image, la procédure est simple car identique à celle utilisée depuis le début de cette thèse. En revanche pour le détecteur par région, cette procédure ne peut pas s'appliquer. Comme dans le cas des classifieurs génériques de l'oracle 2D, les classifieurs des différentes régions sont indépendants. Cela signifie aussi que lors de la détection, chaque classifieur devra être réglé différemment. Il est peu probable que le seuil optimal de détection d'un classifieur corresponde à tous les autres. Cela nous interdit de comparer leur score de sortie directement et de fabriquer une courbe globale comme précédemment.

Pour résoudre ce problème nous considérons l'heuristique suivante. Supposons que nous voulions connaître le rappel r_{total} pour la précision p. Pour cette précision, nous faisons la moyenne du nombre de piétons détectés dans chaque zone. À partir de ces informations, il est possible de calculer le rappel correspondant. La formule (5.1) explicite la méthode.

$$r_{total}(1-p) = \frac{\sum_{i=1}^{N} Card(région_i) \cdot r_i(1-p)}{\sum_{i=1}^{N} Card(région_i)} \tag{5.1}$$

où :

- p est le taux de précision désiré,
- r_i est le rappel de la $i^{ème}$ région,
- r_{total} est le rappel du système complet,
- $Card(région_i)$ est le cardinal de l'ensemble des piétons présents dans la $i^{ème}$ région.

Les performances globales des deux détecteurs sont présentées sur la figure 5.15 et les tables 5.2 et 5.3 résument les performances de chaque classifieur entraîné avec des exemples venant de chaque région ou venant de toute l'image.

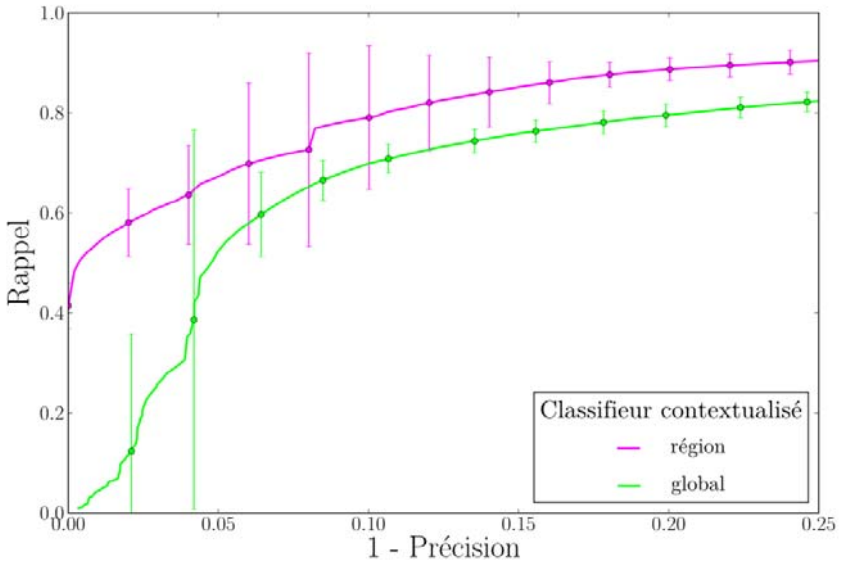


FIGURE 5.15 – Courbes précision-rappel pour toute l'image

TABLE 5.2 – Performances moyennes des classifieurs entraînés avec des exemples venant de chaque région.

	Rappel, (écart type)	Précision	F-Mesure
région 0	0,79 (0,074)	0,90	0,84
région 1	0,98 (0,0064)	0,98	0,98
région 2	0.96 (0,0078)	0,98	0,97
région 3	0,78 (0,021)	0,79	0,78
global	0,85 (0,027)	0,85	0,85

TABLE 5.3 – Performances moyennes des classifieurs entraînés sans les régions, avec des exemples venant de toute l'image.

	Rappel, (écart type)	Précision	F-Mesure
région 0	0,67 (0,064)	0,79	0,73
région 1	0,93 (0,044)	0,97	0,95
région 2	0,97 (0,0074)	0,97	0,97
région 3	0,81 (0,0065)	0,81	0,81
global	0,76 (0,011)	0,85	0,80

Comme attendu nous pouvons voir que le détecteur par région atteint bien de meilleurs résultats que le détecteur global. Pour une même précision de 85%, le rappel du classifieur entraîné à l'aide des régions atteint 85% mais celui du classifieur global n'est que de 76%.

Au travers de ces expériences, nous avons démontré que l'utilisation d'un classifieur global ne donne pas des résultats optimaux. Un modèle global ne permet pas de prendre en compte toutes les spécificités de la scène. Il est donc trop limité et n'est pas capable de discriminer efficacement les piétons dans toutes les parties de l'image. Cela conduit à une saturation lors de l'apprentissage et ce quel que soit le nombre d'exemples avec lesquels est entraîné le classifieur. L'utilisation des régions permet de spécialiser le modèle en fonction de la portion de l'image traitée. En prenant mieux en compte les propriétés locales de la scène, il est possible de mieux contextualiser le détecteur et d'améliorer les performances de ce dernier.

5.5 Conclusion

L'idée de ce chapitre, était de s'intéresser à la sélection des exemples issus du travail de labellisation de l'oracle. En effet, ce dernier n'a aucune connaissance *a priori* sur la pertinence des exemples collectés. Afin d'entraîner un classifieur dans de bonnes conditions, une étape supplémentaire de filtrage de la base est nécessaire.

La première stratégie explorée est la sélection aléatoire des exemples. Très simple à mettre en œuvre, elle a permis de mieux comprendre les implications du choix des exemples sur les performances du classifieur et ainsi de valider plusieurs faits importants.

Tout d'abord, différents apprentissages réalisés à l'aide de différents ratios d'exemples positifs et négatifs ont montré qu'il n'est pas pertinent d'entraîner un classifieur avec l'ensemble de la base. Les ressources nécessaires sont bien supérieures, tandis que les performances ne sont pas meilleures. 2 000 exemples positifs et 8 000 exemples négatifs suffisent pour construire un classifieur contextualisé. D'autre part, ces expérimentations ont mis en évidence le fait que le choix des exemples négatifs est plus important que celui des exemples positifs. Cela est d'ailleurs indirectement confirmé par les autres stratégies de sélection des exemples mises en place par la suite. Le *boostrapping*, technique souvent employée lors des apprentissages génériques pour choisir les exemples négatifs et focaliser l'apprentissage sur les exemples difficiles, apporte également un léger gain de performances pour la contextualisation automatique, fonctionnant avec une base d'apprentissage bruitée. En revanche, nos tentatives pour définir une stratégie de sélection

des exemples positifs n'ont pas été concluantes. Elles visaient à capturer toute la diversité dans l'apparence des exemples de piétons de la base avec le minimum de données. Pour cela, un module de tracking ainsi que l'algorithme du k-means ont été utilisés afin de trouver les exemples redondants. Au vu des résultats obtenus, il semble qu'une stratégie de sélection des exemples positifs ne soit pas indispensable pour la contextualisation.

Ces expériences ont aussi révélé la présence d'une saturation des performances des classifieurs contextualisés. Malgré les différences dans les apprentissages et même avec parfois assez peu d'exemples, les performances sont souvent relativement proches. Une possibilité pour expliquer ce phénomène est la saturation du modèle de représentation des piétons qui est trop faible pour prendre en compte la diversité des apparences des piétons contenues dans la base. Complexifier le modèle peut certes amener un gain des performances mais cela risque également d'engendrer plus de calculs lors de la phase de détection. Nous avons donc opté pour la création de régions et l'entraînement d'un classifieur pour chacune d'entre elles. D'un point de vue global, le modèle est plus complexe et s'adapte mieux aux spécificités de telles ou telles portions de l'image. D'un point de vue local, la complexité est la même et aucun calcul supplémentaire n'est requis pendant la détection.

Les résultats obtenus montrent que les régions permettent un gain net des performances, en particulier dans les régions où les piétons sont fortement déformés par la perspective. Ainsi cela semble valider le fait que la saturation provenait de la limitation du modèle utilisé. Le découpage en régions et leur utilisation dans le processus de contextualisation a été publié dans [Chesnais *et al.*, 2013].

Dans le chapitre suivant, d'autres aspects de la contextualisation sont abordés. Pour le moment, les travaux se sont concentrés sur la partie apprentissage. Comme étudié dans le chapitre concernant la détection de piétons (*cf* chapitre 2), un détecteur de piéton est constitué de multiples composants comme la définition des hypothèses de recherche. Est-il possible de contextualiser automatiquement ces aspects pour encore améliorer les performances ? De plus, jusqu'à présent nous avons surtout cherché à obtenir le classifieur avec la meilleure courbe de précision-rappel. Or dans une application concrète et contextualisée, il est primordial de choisir automatiquement le point de fonctionnement optimal du système. Cela est encore plus impératif lors de l'utilisation des régions car il faut désormais régler plusieurs classifieurs indépendants.

Partie II : Travaux réalisés

6

Application à un système automatique de détection de piétons

Sommaire

6.1 Introduction 138
6.2 Hypothèses de recherche 138
6.2.1 Zones vides 138
6.2.2 Perspectives 141
6.2.2.1 Exploitation de la densité 142
6.2.2.2 Mise à jour de la carte de densité 143
6.2.2.3 Autocalibrage d'une caméra fixe 143
6.3 Intégration de l'oracle dans le détecteur 144
6.3.1 Étude du bouclage de la procédure de contextualisation 145
6.3.2 Mise à jour du classifieur 147
6.3.2.1 Apprentissage online, exemple du boosting 147
6.3.2.2 Implémentation 150
6.3.2.3 Comparaison Boosting Offline / Online 152
6.4 Réglage automatique du seuil de détection 154
6.4.1 Estimation de seuils locaux 154
6.4.2 Estimation d'un seuil global 155
6.4.2.1 Principe 155
6.4.2.2 Implémentation 158
6.4.2.3 Évaluations 158
6.4.3 Perspectives : algorithmes d'apprentissage asymétriques 162
6.5 Conclusion 163

6.1 Introduction

Dans les chapitres précédents, diverses méthodes ont été proposées afin de construire un classifieur adapté à la scène de manière automatique. Ces approches passent par l'utilisation d'un oracle puis par l'apprentissage des exemples que ce dernier a collectés. Comme déjà mentionné dans le chapitre 2.3, le classifieur est le composant du détecteur qui accepte un vecteur descripteur en entrée et qui en déduit l'appartenance ou non de l'image à la classe des piétons. Un détecteur de piétons est composé de nombreuses autres briques. Trois d'entre elles vont être plus particulièrement étudiées ici.

Ce chapitre est plus prospectif que les précédents. Il propose un système automatique de détection de piétons dans un contexte de vidéosurveillance en essayant de prendre en compte les différentes étapes, de l'installation à l'exploitation du système. La partie 6.2 aborde le problème de la définition des hypothèses de recherche à l'aide de zones d'intérêt, en vue de la création d'un parcours optimal du classifieur dans l'image. Un parcours est optimal quand le classifieur teste le minimum d'hypothèses, tout en maintenant des performances maximales. Puis le paragraphe 6.3 explore différentes méthodes pour intégrer l'oracle dans le détecteur. Enfin la partie 6.4 développe une stratégie pour régler automatiquement le classifieur. En effet, seule la construction du meilleur classifieur contextualisé possible a été évoquée jusqu'à présent, mais pour être efficace il doit être bien réglé.

6.2 Hypothèses de recherche

Un point très important à bien appréhender lors de la construction d'un détecteur est la procédure de création des hypothèses de recherche, c'est-à-dire des boîtes qui vont être soumises au classifieur. Si celles-ci sont trop nombreuses le système sera lent, si elles sont en quantité insuffisante le détecteur ne pourra pas localiser précisément les objets ou pire ne reconnaîtra pas tous les piétons.

L'objectif de cette partie est de comprendre la logique et la structure de la scène afin de définir les zones d'intérêt c'est-à-dire celles qui peuvent accueillir des piétons. Cette connaissance est ensuite intégrée dans le détecteur afin d'optimiser le nombre, la position et la forme des boîtes utilisées.

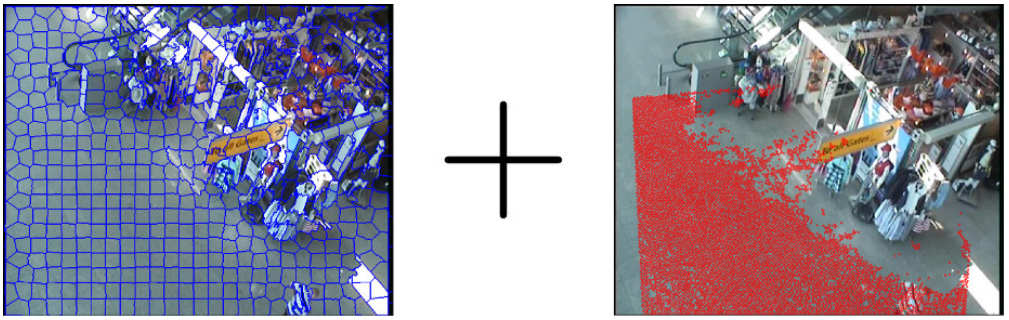
6.2.1 Zones vides

Lors de la détection, toutes les parties d'une image ne sont pas pertinentes. Par exemple, certaines zones sont obstruées par des bâtiments, des cloisons, du mobilier et ne sont donc pas accessibles. Parcourir ces zones à la recherche de piétons, ne peut que détériorer les performances du système. Cela augmente non seulement le temps de calcul mais aussi le nombre de faux positifs car, pour un taux de faux positifs donné, plus le nombre d'hypothèses évaluées est important, plus le risque de faire une erreur est grand et plus les fausses détections sont nombreuses.

La première étape pour détecter les zones vides est d'identifier les différentes parties constituant la scène. Généralement dans une même zone les propriétés visuelles comme la couleur, l'éclairage ou la texture sont homogènes, tandis que la plupart du temps une séparation est visible entre deux zones différentes. Par exemple, il n'est pas rare que le

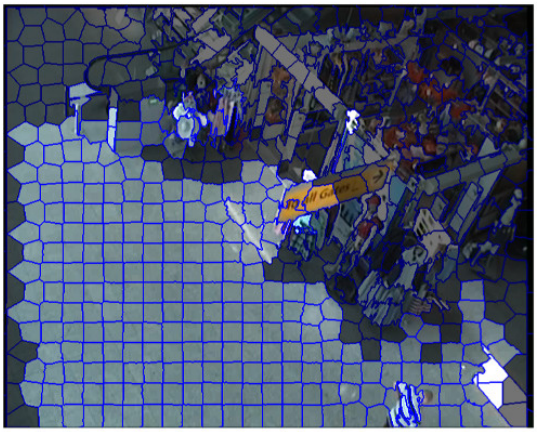
sol et le mobilier aient des couleurs différentes. Comme souvent en analyse d'image, nous commençons par segmenter une image de la scène pour trouver les différents composants de cette dernière. Nous utilisons l'algorithme normal de superpixels SLIC, présenté dans le paragraphe 5.4.2.1 (page 126). Dans nos expérimentations nous avons choisi d'utiliser une grille de superpixels faisant entre 20 et 25 unités de côté.

Les exemples positifs fournis par un oracle sont ensuite repris. Pour cette expérience, l'oracle 3D (*cf* partie 4.4, page 95) a été utilisé. Un exemple est représenté par la position des pieds du piéton au sol. Si un superpixel comporte au moins un exemple alors la zone est considérée comme pouvant accueillir des piétons. Dans le cas contraire cette zone est ignorée lors du parcours dans l'image (*cf* figure 6.1).



(a) - Segmentation superpixel

(b) - Positions 3D fournies par l'oracle



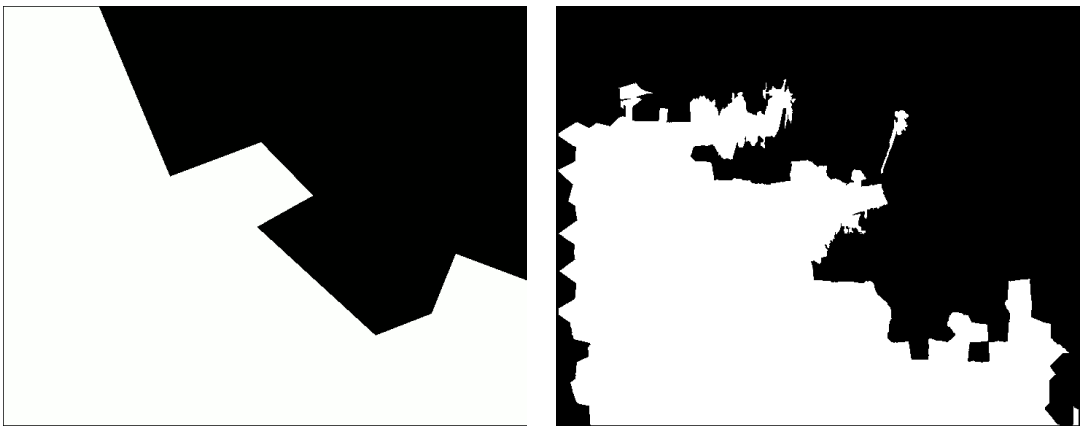
(c) - Détections des zones intéressantes
et élimination des zones vides

FIGURE 6.1 – Exemple de cartographie générée automatiquement. (a) Segmentation d'une image de la scène par superpixel. Les frontières des clusters sont représentées par les lignes bleues. (b) Positions des pieds de tous les piétons détectés par l'oracle 3D. (c) Obtention des zones d'intérêt et suppression des zones vides grâce au regroupement des informations de (a) et (b). Les zones claires sont celles contenant des piétons tandis que les zones foncées sont vides.

Un mécanisme similaire de création de zones d'intérêt existe dans le système, basé sur les *classifier grids*, proposé par [Stalder *et al.*, 2009b]. Un classifieur online est initialisé dès qu'un piéton est détecté dans une nouvelle zone par leur oracle qui est composé entre autres d'un classifieur offline. Contrairement à notre approche, ce mécanisme nécessite

que les zones vides soient parcourues en permanence par le classifieur offline afin de mettre le détecteur à jour. Il n'y a donc aucun gain de temps lors de la phase de détection et toutes les hypothèses sont au minimum évaluées soit par l'oracle, soit par un classifieur online.

Pour mesurer l'impact de la prise en compte des zones vides par le détecteur, nous comparons les performances de divers classifieurs contextualisés, certains utilisant ces zones et un autre non. Afin de valider notre algorithme de création automatique des zones, l'expérience est réalisée à la fois avec les zones fabriquées à la main par un humain et avec celles obtenues à l'aide de notre approche. La figure 6.2 montre les zones obtenues manuellement et celles obtenues automatiquement. Cinq classifieurs sont entraînés pour chacune des catégories. 2 000 exemples positifs et 8 000 exemples négatifs sont appris par chaque classifieur. Lors de l'apprentissage des classifieurs, les exemples négatifs situés dans une zone vide sont ignorés.



(a) Zones obtenues manuellement (b) Zones obtenues automatiquement

FIGURE 6.2 – Zones vides (en noir) obtenues manuellement ou automatiquement

Les courbes 6.3 présentent les résultats de l'expérience et indiquent que l'utilisation des zones vides améliore les performances du détecteur. Cela n'est pas étonnant car comme déjà signalé, en testant moins d'hypothèses, le classifieur a moins de risque de se tromper. Il teste néanmoins les boîtes qui contiennent les piétons. Ainsi son rappel ne faiblit pas et sa précision augmente mécaniquement.

La table 6.1 présente les temps de calcul du balayage dans trois cas différents et le nombre d'hypothèses testées (échantillonnées selon une grille régulière de pas 2 cm dans le plan du sol). Le temps des post-traitements, c'est-à-dire du regroupement des boîtes, est inférieur à 5 millisecondes et est négligeable par rapport à celui du parcours du détecteur dans l'image. Logiquement, le temps de calcul est proportionnel au nombre de boîtes. Les classifieurs utilisant les zones vides sont donc plus rapides. Sur cette séquence, l'utilisation de zones vides permet de traiter une image en moins de 100 millisecondes et de rendre le système compatible avec le temps réel.

La différence principale entre l'algorithme de création des zones et l'humain se situe au niveau des frontières. Une personne aura tendance à préférer les lignes droites plus simples alors que cela n'est pas forcément optimal. Néanmoins, qualitativement, les différences entre les deux sont minimes, mais l'avantage va à notre algorithme automatique sur cette séquence.

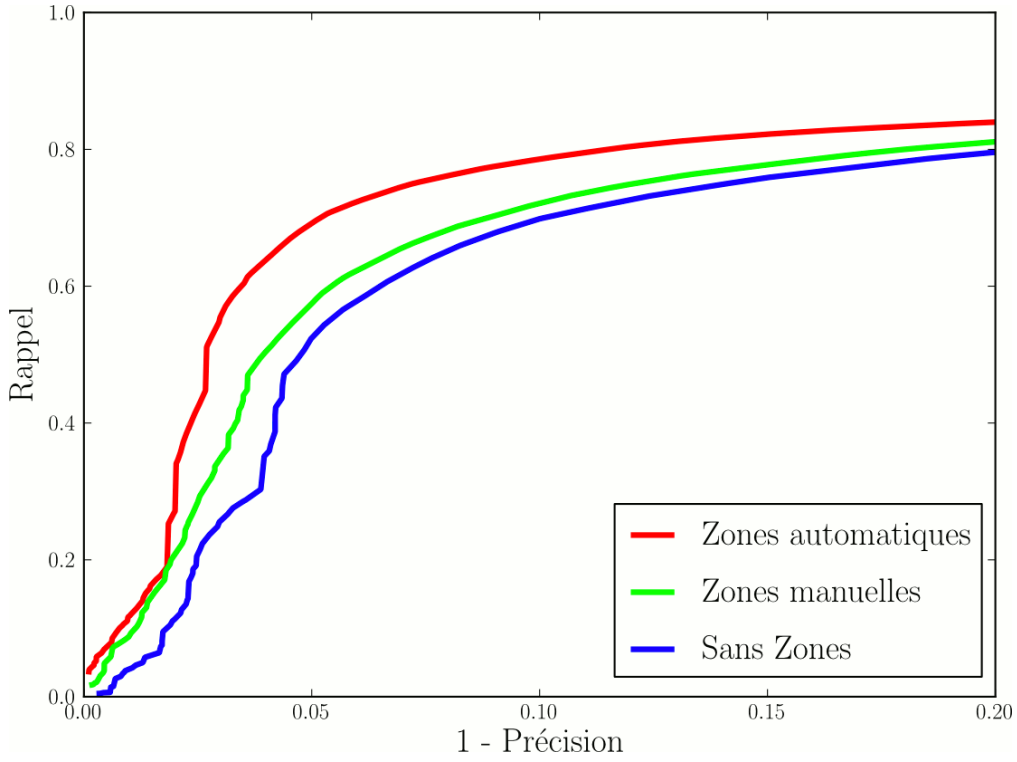


FIGURE 6.3 – Courbes précision-rappel des trois systèmes. La rouge correspond au système prenant en compte les zones vides créées automatiquement par notre algorithme. La verte correspond au système prenant en compte les zones vides créées manuellement et la bleue au système n'utilisant pas les zones.

TABLE 6.1 – Nombre d'hypothèses et temps de calcul moyen pour traiter une image

Zones vides	non	oui (automatique)	oui (manuelle)
Nombre d'hypothèses	16 661	9 895	10 174
Temps de calcul moyen (en ms)	122	74	77

6.2.2 Perspectives

Les approches suivantes nous semblent être une continuation possible de nos travaux sur le parcours du classifieur. Les premières sont basées sur l'exploitation de la carte de densité des piétions dans la scène et les secondes s'intéressent à l'autocalibrage d'une caméra fixe. Nous n'avons pas testé ces méthodes.

6.2.2.1 Exploitation de la densité

La carte des hypothèses créée jusqu'à présent ne fournit qu'une information binaire : y a-t-il eu un piéton détecté dans cette zone pendant la phase d'apprentissage ou pas ? Or nous possédons des données plus riches puisque nous disposons du nombre de piétons détectés dans chaque zone. L'idéal serait de pouvoir utiliser cette information de densité. En effet, il semble plus probable qu'un piéton soit présent dans une zone qui en a déjà vus plusieurs, plutôt que dans une zone où une seule personne a été détectée. La figure 6.4 présente la densité estimée dans la scène, les parties noires indiquent les endroits pour lesquels il n'y a pas de données, les parties jaunes les endroits où il y a eu peu de piétons et les parties rouges les endroits où beaucoup de piétons ont été détectés par l'oracle.

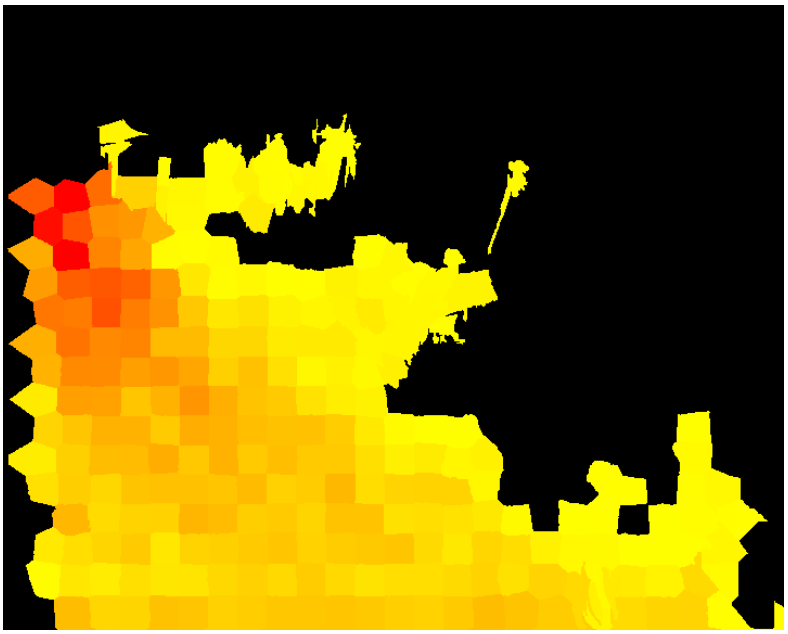


FIGURE 6.4 – Carte de densité estimée des piétons dans la scène. (jaune : densité faible, rouge : densité élevée, noir : pas de piétons)

La méthode de parcours basée sur les fenêtres glissantes (*cf* paragraphe 2.3.3.3, page 35) ne permet pas de prendre en compte la probabilité de présence d'un objet. L'idée serait alors d'utiliser une approche par tirage particulaire (*cf* paragraphe 2.3.3.3, page 36). En échantillonnant lors du premier étage, non plus aléatoirement comme dans l'article de [Gualdi *et al.*, 2010] mais en tenant compte de l'estimation de la densité, il serait possible d'adapter au mieux le parcours du détecteur dans la scène. L'un des avantages est qu'il serait alors possible de tirer moins de particules dans les zones presque vides plutôt que d'y accorder autant d'importance qu'à la zone la plus peuplée. Les zones presque vides étant parfois des superpixels qui ne contiennent que très peu de détections dont de nombreux faux positifs.

6.2.2.2 Mise à jour de la carte de densité

La mise à jour de la carte de densité est une autre extension possible à notre système. En effet, si seules les zones ayant déjà contenu un piéton sont testées alors les zones vides au début de la séquence resteront vides à jamais. Cela peut-être problématique si la scène change. En échantillonnant au minimum lors de la phase de détection les zones vides, il pourrait être possible de détecter une incursion éventuelle dans la zone et de prendre en compte cette nouvelle information par la suite.

6.2.2.3 Autocalibrage d'une caméra fixe

Nous avons déjà évoqué précédemment et plus particulièrement dans le chapitre 4 que la détection de piétons est plus performante si elle est contrainte par une information 3D. Grâce à cela, le système connaît précisément la taille, l'orientation d'un objet en fonction de sa position dans le repère monde et aussi dans l'image. Souvent cette information est calculée lors de la phase de calibration de la caméra. Ainsi le système peut se permettre de tester beaucoup moins d'hypothèses dans l'image. Néanmoins, cette opération de calibration est lourde et coûteuse puisqu'il faut traiter indépendamment chaque caméra. Parfois et notamment dans un réseau de caméras à champs joints, il peut également être intéressant de connaître la position des caméras les unes par rapport aux autres.

Dans le cas de caméras fixes, deux familles de méthodes existent pour trouver de manière automatique les caractéristiques des objets en fonction de leur position dans l'image.

Les premières, appelées calibration faible, estiment directement pour chaque point de l'image la taille des objets, bien que les paramètres internes et externes de la caméra restent inconnus.

Dans leurs travaux, [Wang et Wang, 2011] se servent des sorties d'un détecteur de piétons 2D non contextualisé. La taille et la position de ces détections fournissent une première indication sur la répartition spatiale des tailles des objets dans l'image. Il est en effet très peu probable qu'une détection de 100 pixels puisse exister à côté d'une de 10 pixels. Cette répartition est modélisée à l'aide d'une gaussienne. Le but ici n'est pas de trouver les hypothèses les plus proches de la réalité mais de supprimer celles qui sont manifestement inadéquates.

[Hoiem *et al.*, 2005, 2008] tentent de déduire le maximum de propriétés de la scène à l'aide d'une seule image. Le but étant, à partir d'un ensemble de détections, d'estimer les principales caractéristiques de la scène comme le plan du sol, les plans verticaux, l'horizon et le ciel, la hauteur et l'angle de la caméra puis de s'en servir pour affiner les détections. Il est probable, du fait des hypothèses effectuées que cette méthode ne fonctionne correctement qu'en extérieur.

[Breitenstein *et al.*, 2008] proposent de construire grossièrement la géométrie 3D de la scène. Ils commencent par discrétiser l'image à l'aide d'une grille régulière 2D. Un détecteur de piétons 2D fournit la boîte des piétons de chaque case, ce qui permet d'estimer la taille la plus probable des piétons d'une case et d'en déduire le relief (la profondeur dans le repère monde) de la scène observée à cet endroit. Les données du détecteur étant forcément bruitées, la méthode, pour augmenter sa robustesse, modélise de plus les interactions entre cases adjacentes et accumule les données pour mettre le

modèle à jour au fur et à mesure. Les estimations une fois affinées sont envoyées au détecteur afin qu'il puisse tester moins d'hypothèses. L'avantage de cette approche est qu'elle nécessite peu d'hypothèses sur la scène. En particulier les piétons peuvent évoluer sur plusieurs niveaux et ne sont plus contraints sur un plan principal.

Les secondes méthodes, appelées calibration forte, visent à automatiser les procédures de calibration en estimant les paramètres internes et externes de la caméra. Généralement, ces algorithmes sont basés sur la recherche des points de fuite. Les premiers algorithmes (par exemple [Rother, 2000]) proposés dans la littérature recherchent les segments dans l'image. Ces derniers sont ensuite prolongés par des droites et leurs points de rencontre, c'est-à-dire les points de fuite, sont déterminés à l'aide d'un RANSAC [Fischler et Bolles, 1981].

Cependant il n'y a pas toujours suffisamment de segments dans une scène. Une autre approche [Lv *et al.*, 2006] pour trouver les points de fuite consiste à segmenter précisément les objets de la scène et notamment les piétons. Deux droites qui passent soit par les pieds soit par les têtes doivent converger vers un point de fuite à condition qu'il s'agisse à chaque fois de la même personne. Il est ainsi possible de trouver les points de fuite en accumulant les positions d'un même piéton dans la scène. [Liu *et al.*, 2011] ont amélioré cette approche. Celle-ci utilise la taille moyenne connue des humains, ce qui lui permet de fonctionner avec des piétons différents. Cette méthode plus simple à mettre en œuvre est particulièrement adaptée aux caméras de vidéosurveillance.

Dans cette thèse, toutes les étapes de calibration ont été réalisées manuellement. Les approches automatiques de calibration qui estiment les paramètres de la caméra semblent plus intéressantes car plus précises. Une fois l'équation du plan du sol connue, il est possible de précalculer les positions admissibles des piétons comme évoqué dans la partie sur les fenêtres glissantes 2.3.3.3 (page 35). Un travail supplémentaire est nécessaire pour valider l'utilisation d'une méthode d'autocalibrage dans le cadre de la contextualisation et ainsi pouvoir bénéficier de l'oracle 3D plus facilement. Néanmoins aucune étape ne nous semble rédhibitoire puisque ces approches ont déjà été validées dans des systèmes de détection.

6.3 Intégration de l'oracle dans le détecteur

Les oracles présentés dans le chapitre 4 ont pour but d'extraire, avec le moins d'erreur possible, des exemples pertinents positifs et négatifs en provenance de la scène. Dans le chapitre 5 nous avons cherché la meilleure manière d'entraîner un classifieur contextualisé à l'aide de ces données très spécifiques. Néanmoins, l'intégration dans le système de l'oracle a été peu évoquée. Le but de cette partie est de s'intéresser aux interactions entre l'oracle et le classifieur contextualisé. Dans un premier temps, nous étudierons si boucler le processus de contextualisation engendre de meilleurs résultats pour le classifieur final. Dans un deuxième temps, nous verrons si la classification online améliore les performances du système.

6.3.1 Étude du bouclage de la procédure de contextualisation

Le système de contextualisation présenté dans cette thèse permet de produire des classifieurs spécialement adaptés à la scène. Pour le moment, seul le détecteur contextualisé en bénéficie. Or il pourrait être intéressant d'utiliser aussi un classifieur contextualisé dans l'oracle afin d'augmenter son rappel et sa précision dans le but de constituer une meilleure base d'apprentissage.

Il est possible de réaliser la procédure de contextualisation plusieurs fois de suite. Dans un premier temps, l'oracle générique labellise une petite portion de vidéo de 2 000 images. Un classifieur contextualisé est créé à partir de ces exemples, puis est intégré dans l'oracle à la place du classifieur générique. Dans un deuxième temps, cet oracle contextualisé labellise 2 000 nouvelles images de la vidéo puis ajoute ces exemples aux précédents. Une nouvelle itération débute alors par la création d'un nouveau classifieur.

L'expérience a été réalisée sur la séquence ViCoMo 1 et utilise un oracle 3D. Les 10 000 premières images servent à annoter une base d'apprentissage et les images numérotées de 15 000 à 16 000 constituent la base de test. Chaque classifieur a été entraîné avec 2 000 exemples positifs et 8 000 exemples négatifs. Le classifieur de l'itération i utilise lors de son entraînement les exemples labellisés par tous les oracles précédents. Le classifieur 4 possède donc dans sa base d'apprentisage des exemples issus des images 0 à 8 000.

Pour mesurer les impacts du bouclage sur la contextualisation, un classifieur témoin est fabriqué et est utilisé comme étalon. Il est entraîné avec l'aide de l'oracle générique ayant labellisé les 10 000 images. Le classifieur contextualisé témoin et le classifieur contextualisé après l'itération 4 ont donc été entraînés avec la même quantité d'images labellisées.

TABLE 6.2 – Performances des oracles itératifs et de l'oracle témoin. Les résultats sont donnés pour chaque tranche de 2 000 images et de manière globale pour les 10 000 images.

	Rappel	Précision	F-Mesure
Oracle itératif			
itération 0 : 0 - 2 000	0,60	0,91	0,72
itération 1 : 2 000 - 4 000	0,67	0,90	0,77
itération 2 : 4 000 - 6 000	0,68	0,88	0,77
itération 3 : 6 000 - 8 000	0,84	0,80	0,82
itération 4 : 8 000 - 10 000	0,78	0,76	0,77
images : 0 - 10 000	0.70	0.85	0.77
Oracle témoin			
images : 0 - 2 000	0,60	0,91	0,72
images : 2 000 - 4 000	0,62	0,94	0,75
images : 4 000 - 6 000	0,60	0,95	0,74
images : 6 000 - 8 000	0,76	0,86	0,81
images : 8 000 - 10 000	0,66	0,90	0,76
images : 0 - 10 000	0,64	0,92	0,75

La table 6.2 présente les points de fonctionnement des différents oracles créés à l'aide du bouclage, ainsi que celui de l'oracle témoin. Les statistiques sont calculées pour tous les oracles à l'aide de 2 000 images, d'où le découpage en tranches. Ceci permet de comparer les performances d'un oracle itératif et de l'oracle témoin en utilisant les mêmes images de test. En effet, des images plus ou moins difficiles à traiter peuvent fausser les comparaisons entre des tranches différentes. La dernière ligne correspond aux performances sur toute la séquence. Ce tableau montre que l'oracle contextualisé a un meilleur rappel que l'oracle générique. Par contre cela se fait au détriment de sa précision qui diminue à chaque itération. Le système dérive.

La figure 6.5 présente les performances des classifieurs contextualisés successifs obtenus. La table 6.3 indique les points de fonctionnement, optimaux au sens de la F-Mesure, des différents classifieurs contextualisés obtenus.

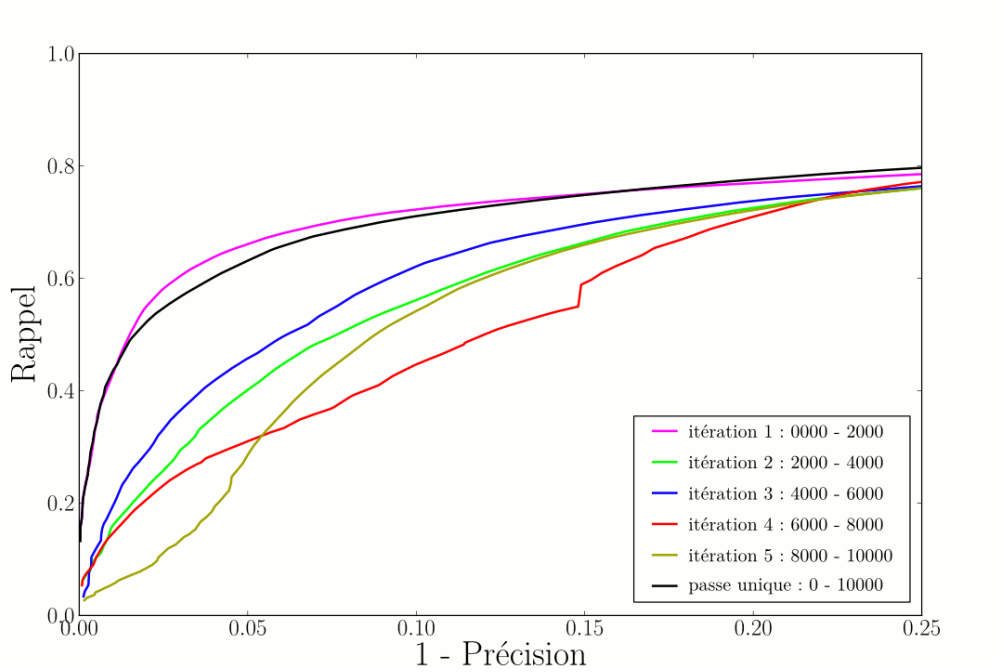


FIGURE 6.5 – Effet du bouclage de la procédure de contextualisation sur les performances d'un classifieur contextualisé.

Tous ces résultats indiquent que le bouclage de la procédure de contextualisation détériore les résultats. Lors de la première itération, le classifieur contextualisé a les mêmes performances que le classifieur témoin. Lors des itérations suivantes, bien que le rappel de l'oracle augmente, sa précision diminue. Cela signifie que de plus en plus d'éléments du fond sont intégrés dans la base des exemples positifs. Les erreurs de l'oracle sont alors amplifiées par les contextualisations successives et les performances des classifieurs contextualisés décroissent. Cette expérience montre à nouveau qu'un oracle précis est plus important qu'un oracle avec un fort rappel.

TABLE 6.3 – Performances des classifieurs contextualisés.

	Rappel, (écart type)	Précision	F-Mesure
Classifieurs itératifs			
itération 0 : 0 - 2 000	0,73 (0,051)	0,89	0,80
itération 1 : 2 000 - 4 000	0,73 (0,051)	0,79	0,76
itération 2 : 4 000 - 6 000	0,72 (0,079)	0,82	0,77
itération 3 : 6 000 - 8 000	0,76 (0,075)	0,77	0,76
itération 4 : 8 000 - 10 000	0,74 (0,062)	0,78	0,76
Classifieur témoin			
images : 0 - 10 000	0,74 (0,020)	0,86	0,80

6.3.2 Mise à jour du classifieur

Jusqu'à présent, nous avons étudié de manière exclusive l'apprentissage offline. Dans ce cas la base est intégralement connue pendant la phase d'entraînement. L'algorithme d'apprentissage a donc une vue d'ensemble du problème et peut ainsi plus facilement définir la séparation entre les classes. Cette approche paraît plus simple à appréhender mais elle n'est pas optimale dans plusieurs cas. Par exemple, si les données arrivent en flux continu comme dans le cas d'une vidéo, il est contraignant d'attendre la fin de celle-ci pour constituer la base. D'autre part cette dernière peut être trop volumineuse et dépasser la capacité mémoire de l'unité de traitement, rendant l'apprentissage impossible. Enfin le classifieur est statique. Cela signifie que si le besoin de le mettre à jour se fait sentir, un apprentissage intégral est à nouveau nécessaire.

6.3.2.1 Apprentissage online, exemple du boosting

Pour pallier ces problèmes, des méthodes d'apprentissage online (ou en ligne) sont apparues. Elles consistent à traiter un (ou plusieurs suivant les définitions) exemple(s) pour mettre à jour un classifieur. L'exemple est ensuite jeté et n'occupe plus de place dans l'ordinateur. Les avantages de cette approche sont simples. La base d'apprentissage n'a plus besoin de résider en permanence dans la mémoire, ce qui permet d'entraîner le classifieur sur de grands ensembles. D'autre part le classifieur peut théoriquement être mis à jour en permanence sans avoir besoin de tout réapprendre. Il peut donc être mieux contextualisé si la scène change au cours du temps. L'inconvénient majeur, outre la difficulté pour les algorithmes d'avoir une vision globale du problème avec quelques observations, est l'obtention des exemples. Les algorithmes online sont souvent couplés avec une méthode d'apprentissage semi-supervisée, ce qui apporte de nombreuses complications par rapport aux méthodes offline et supervisées plus classiques. D'autre part contrairement aux algorithmes offline très rigides, les approches online sont très souples, ce qui peut poser problème si la base est bruitée. En effet, le classifieur aura du mal à converger vers une bonne solution et les performances risquent d'être médiocres. L'un des rôles de la méthode de supervision est donc de contraindre suffisamment le problème sans pour autant restreindre l'adaptabilité de la méthode.

Dans notre cadre applicatif, utilisant déjà un oracle qui peut fournir des observations image par image, le choix d'une méthode online plutôt que offline peut sembler pertinent.

De la même façon que pour le boosting offline, il existe de nombreuses variantes du boosting online. La version initialement proposée par [Oza et Russell, 2001], reprend le formalisme des travaux de [Freund et Schapire, 1999], c'est-à-dire du boosting sans sélection de composantes. Quelques années plus tard, [Grabner et Bischof, 2006] popularise cet algorithme dans la communauté de la vision par ordinateur en intégrant la sélection des composantes comme [Viola et Jones, 2001b] dans le cas offline. L'algorithme 9 présente la procédure du boosting online proposé par [Grabner et Bischof, 2006].

Le boosting online a été utilisé dans les principaux domaines de la vision pouvant nécessiter l'emploi d'un classifieur : détection d'objets, soustraction de fond... Mais c'est probablement dans le cadre du tracking qu'il a été le plus employé pour créer des modèles d'apparence des objets suivis [Stalder *et al.*, 2009a; Breitenstein *et al.*, 2011].

Les algorithmes d'apprentissage du boosting offline et online, bien que similaires dans l'esprit, présentent plusieurs divergences.

Premièrement, dans le boosting online, il n'y a plus de notion de round (ou d'itération). Par construction, l'algorithme ne peut donc plus modifier le nombre de classifieurs faibles présents dans le classifieur fort. Lors de l'apparition d'un nouvel exemple, tous les classifieurs faibles présents sont réentraînés pour tenir compte des informations disponibles.

Deuxièmement, dans un algorithme de boosting online, il est obligatoire d'utiliser des classifieurs faibles online. Les *decision stumps* ne sont donc plus appropriés. Les classifieurs faibles proposés par [Grabner et Bischof, 2006] utilisent des gaussiennes 1D. Pour une composante du descripteur, il faut entraîner deux gaussiennes, g_{μ_+,σ_+} et g_{μ_-,σ_-}, une pour les exemples positifs et une pour les exemples négatifs. L'entraînement consiste à calculer en ligne la moyenne et la variance des valeurs sur cette composante. Les deux gaussiennes modélisent $p(y = +1|x)$ et $p(y = -1|x)$, c'est-à-dire les probabilités que l'observation appartienne à la classe des positifs ou des négatifs connaissant la valeur de telle composante du descripteur. L'algorithme proposé par [Grabner et Bischof, 2006] étant une version discrète, ce classifieur faible renvoie $+1$ si $g_{\mu_+,\sigma_+}(x) > g_{\mu_-,\sigma_-}(x)$ et -1 sinon. La figure 6.6 présente le fonctionnement de ce classifieur faible.

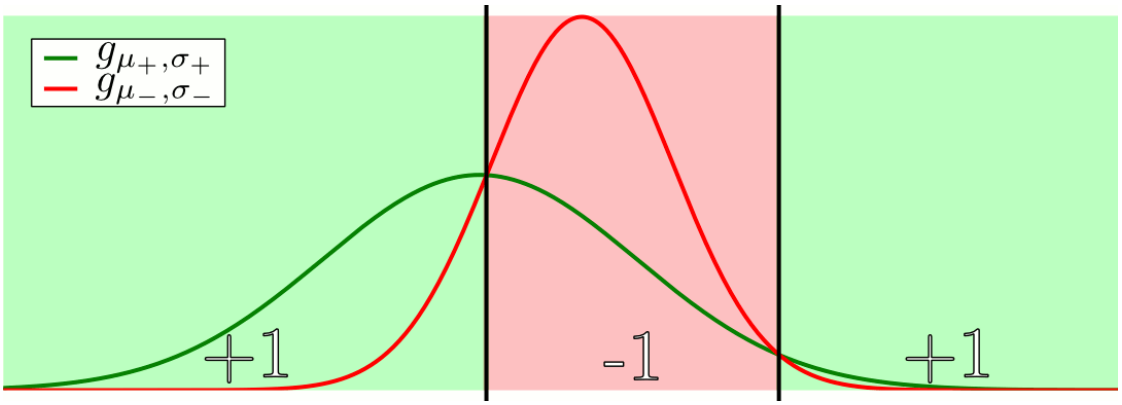


FIGURE 6.6 – Principe de fonctionnement du classifieur faible online à base de gaussiennes.

La troisième différence entre le boosting online et le boosting offline concerne la gestion du poids des exemples. Dans le cas du boosting offline, la distribution du poids des exemples sert de mémoire à l'algorithme. Plus le poids d'un exemple est élevé et moins il a été bien classé par les classifieurs faibles déjà sélectionnés. Il est donc important pour les classifieurs faibles futurs de bien le prendre en compte. Dans le cas du boosting online, les exemples n'étant pas tous connus lors de l'entraînement, il n'est plus possible de s'appuyer sur le même mécanisme. Le boosting online utilise donc un système légèrement différent. L'exemple courant possède un poids appelé importance. Plus son importance est grande et plus il va pouvoir influer sur un classifieur faible. L'importance est ensuite réévaluée après l'entraînement du classifieur faible. Elle est diminuée si ce dernier arrive à retrouver le label correct de l'exemple et augmentée sinon. Cette modification tient également compte de la pertinence du classifieur faible. Celle-ci correspond à l'erreur commise par ce classifieur sur tous les exemples traités depuis le début.

Enfin, un algorithme online destiné au temps réel doit être rapide à entraîner. Or le boosting avec sélection de composantes impose d'entraîner tous les classifieurs faibles potentiels et de choisir le meilleur pour mettre à jour le classifieur fort. Cela est bien sûr très coûteux en temps de calcul car dans le cas du boosting online ceci devrait être effectué pour chaque nouvel exemple. [Grabner et Bischof, 2006] ont donc introduit la notion de sélecteur (figure 6.7) qui s'intercale entre les notions de classifieur fort et de classifieur faible. Un sélecteur est un conteneur de classifieurs faibles qui ne possède qu'un échantillon parmi tous les classifieurs faibles possibles. Lors de la phase d'apprentissage, plutôt que d'entraîner tous les classifieurs faibles possibles à chaque nouvel exemple et pour chaque position dans le classifieur fort, seuls les classifieurs faibles présents dans les sélecteurs sont évalués. Le meilleur classifieur de chaque sélecteur est intégré dans le classifieur fort. Ce sous-échantillonnage des classifieurs faibles limite fortement la quantité de calculs nécessaire. Certains classifieurs faibles présents dans le sélecteur peuvent ne pas être adaptés au problème traité. Cela est d'autant plus gênant que tous les classifieurs faibles ne sont pas évalués. Dans ce cas, le classifieur faible le moins pertinent est supprimé et remplacé par un nouveau. Comme souvent, lorsqu'une entité est mise à jour en ligne, il est important de ne pas négliger la robustesse face à l'adaptabilité. Dans le cas des sélecteurs, un remplacement des classifieurs faibles après l'entraînement de nombreux exemples limite l'adaptabilité du classifieur mais impose une certaine rigidité au système. Les classifieurs faibles ont le temps d'être correctement entraînés et le choix du remplacement est motivé. Dans le cas contraire, le classifieur faible nouvellement promu n'a pas le temps d'être entraîné et risque de disparaître à l'itération suivante, même si après un certain temps, il aurait été parfaitement adapté au problème traité.

La figure 6.7 présente l'architecture de l'algorithme de boosting online avec sélection de composantes proposé par [Grabner et Bischof, 2006].

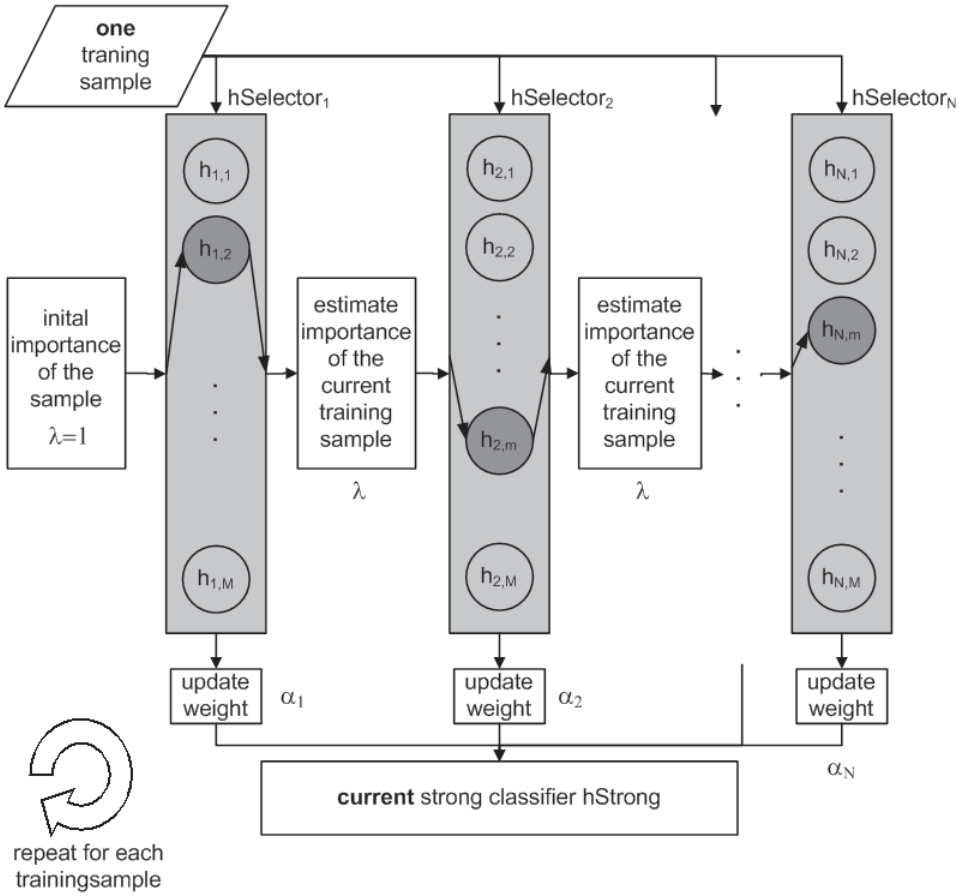


FIGURE 6.7 – Architecture de l'algorithme de boosting online avec sélection de composantes (extrait de [Grabner et Bischof, 2006])

6.3.2.2 Implémentation

L'implémentation du boosting online retenue repose sur la version de [Grabner et Bischof, 2006]. Il s'agit donc d'un Adaboost discret avec sélection de composantes. Tous les classifieurs offline entraînés jusqu'à présent comportent 400 classifieurs faibles. Pour réaliser une comparaison cohérente entre les versions offline et online du boosting, nous avons décidé de travailler avec des classifieurs online munis de 400 sélecteurs. Chaque sélecteur est indépendant et contient 200 classifieurs faibles tirés aléatoirement. Afin de ne pas perturber l'apprentissage, la procédure de remplacement des classifieurs faibles au sein des sélecteurs est effectuée tous les 1 000 exemples.

Algorithme 9: Adaboost online discret avec sélection de composantes

Entrées :
Un exemple d'apprentissage $\{(x, y), x \in \mathbb{X}, y \in \mathbb{Y} = \{-1, +1\}\}$,
n le nombre de sélecteurs,
m le nombre de classifieurs faibles par sélecteur,
un ensemble $\mathbb{H}_f$ de classifieurs faibles.

Sorties :
Un classifieur fort h

Initialisation :

Initialiser les n sélecteurs en choisissant aléatoirement m classifieurs faibles de $\mathbb{H}_f$: h_t^j avec $1 \leq t \leq n$ et $1 \leq j \leq m$.
Soit $\lambda_{t,j}^{corr}$ et $\lambda_{t,j}^{wrong}$ les coefficients d'estimation de l'erreur associés à chaque classifieur faible, initialisés à 1
Soit $\lambda = 1$, l'importance associée à l'exemple

Algorithme :

pour $t = 1, ..n$ **faire**
  pour $j = 1, ..m$ **faire**
    Entraîner tous les classifieurs faibles du sélecteur t, $h_t^j : \mathbb{X} \to \mathbb{Y}$, en tenant compte de son importance λ
    Estimer l'erreur de chaque classifieur faible :
    si $h_t^j(x) = y$ **alors**
      $\lambda_{t,j}^{corr} = \lambda_{t,j}^{corr} + \lambda$
    sinon
      $\lambda_{t,j}^{wrong} = \lambda_{t,j}^{wrong} + \lambda$
    fin
    $\epsilon_t^j = \frac{\lambda_{t,j}^{wrong}}{\lambda_{t,j}^{corr} + \lambda_{t,j}^{wrong}}$
  fin
  Choisir le classifieur faible, h_t^J, avec l'erreur, ϵ_t^J la plus faible
  Affecter le poids $\alpha_t = 0,5 \ \ln(\frac{1-\epsilon_t^J}{\epsilon_t^J})$ à h_t^J
  Mettre à jour l'importance de l'exemple :
  si $h_t^J(x) = y$ **alors**
    $\lambda = \frac{\lambda}{2(1-\epsilon_t^J)}$
  sinon
    $\lambda = \frac{\lambda}{2.\epsilon_t^J}$
  fin
fin

Le classifieur fort est défini par :

$$h = \sum_t \alpha_t h_t \tag{6.1}$$

6.3.2.3 Comparaison Boosting Offline / Online

Le principal intérêt des approches online réside dans le fait de pouvoir mettre à jour le classifieur facilement et donc de pouvoir le contextualiser dès qu'il y a des changements dans la scène. Par contre, cela exige que l'oracle fonctionne en parallèle du détecteur final ce qui augmente d'autant les temps de calcul.

Pour savoir si les avantages du boosting online compensent ses faiblesses, les expériences suivantes ont été réalisées sur la séquence ViCoMo 1 car c'est elle qui possède la plus longue vérité terrain. Tout d'abord nous avons comparé les performances du boosting online et du boosting offline. Pour cela un protocole expérimental similaire à celui déjà présenté dans les chapitres précédents est utilisé. L'oracle labellise les images 25 000 à 35 000. Un classifieur contextualisé est entraîné grâce au boosting online ou au boosting offline. 8 000 exemples négatifs et 2 000 exemples positifs sont sélectionnés aléatoirement parmi la base construite par l'oracle pour fabriquer un classifieur offline. Par contre, l'intérêt majeur d'un classifieur online étant de pouvoir prendre en compte un nombre très important d'exemples, pour construire ce dernier, tous les exemples fournis par l'oracle ont été utilisés, ce qui représente environ 45 000 exemples positifs et 50 000 exemples négatifs. Le classifieur contextualisé comme l'oracle utilisent la calibration de la caméra et sont donc 3D. Les performances des deux systèmes sont évaluées sur les 20 000 premières images de la séquence, ce qui représente environ 15 minutes de vidéo.

Cette procédure est répétée cinq fois et les résultats moyens sont présentés sur la figure 6.8. Globalement, le classifieur online (courbe verte) a de moins bonnes performances que le classifieur offline (courbe magenta). Ceci s'explique peut-être par le fait que les comparaisons ont été réalisées à l'aide d'une version RealAdaboost en offline et d'une version DiscretAdaboost en online.

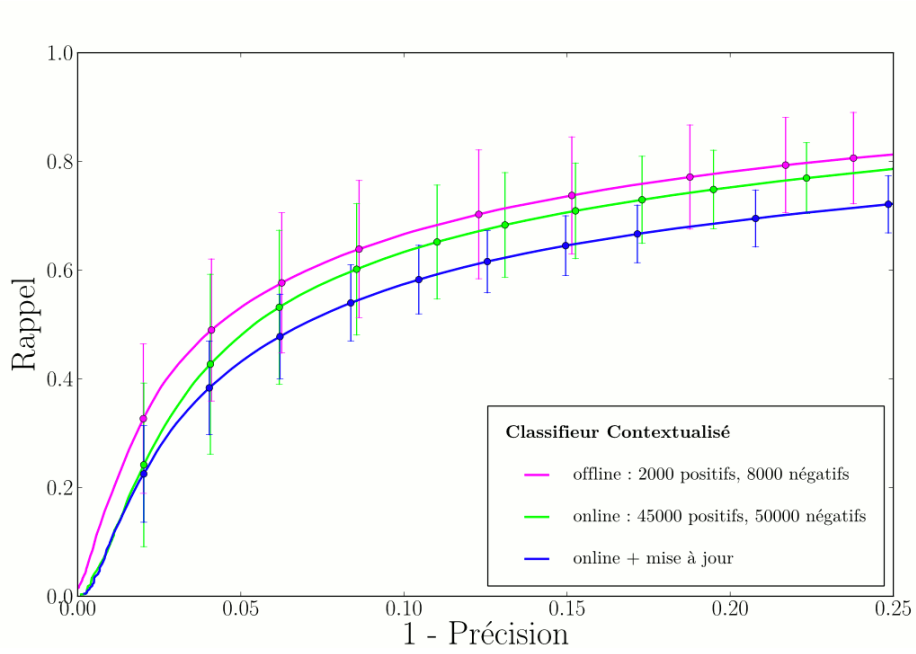


FIGURE 6.8 – Comparaison des performances d'un classifieur offline et d'un classifieur online

Cette expérience ne teste que les performances pures des deux classifieurs. Or l'avantage du classifieur online est de pouvoir être mis à jour en temps réel. Pour savoir si cela peut-être décisif face au classifieur offline, les tests suivants ont été réalisés. Le classifieur online construit précédemment est mis à jour, pendant son évaluation, avec les données de la séquence de test. Concrètement le classifieur online et l'oracle labellisent en permanence la vidéo. Si l'oracle détecte des boîtes non repérées par le classifieur online, avec un seuil de détection fixé à 0, alors le classifieur online apprend ces nouveaux exemples positifs. Les exemples négatifs sont toujours tirés aléatoirement dans la scène aux endroits où il n'y a pas de détection de l'oracle. Autant d'exemples positifs et d'exemples négatifs sont appris à chaque mise à jour.

Les résultats de cette expérience réalisée cinq fois sont présentés sur la courbe bleue de la figure 6.8. La mise à jour du classifieur, demande non seulement plus de calculs, mais elle semble dégrader les performances du système et n'offre pas d'avantage face au classifieur offline. Ce résultat peut s'expliquer par le fait que l'entraînement d'un système online est délicat.

Dans la première expérience, le classifieur online est entraîné de façon non linéaire avec des exemples provenant de nombreuses images espacées dans le temps. Des exemples consécutifs ont donc peu de chance d'être similaires. Ce n'est pas le cas dans la seconde expérience, dans laquelle le classifieur online risque de surapprendre les exemples courants qu'il a du mal à classer correctement et donc de perdre en généralisation. Dans le cas où l'oracle commet des erreurs, le problème est accentué car le classifieur online va apprendre pendant quelques images de nombreuses erreurs qui auraient pu être noyées dans la base d'apprentissage dans le premier cas. Ce phénomène peut être limité en utilisant une approche incrémentale. Les exemples de la scène sont stockés puis appris à intervalles de temps réguliers.

L'autre problème soulevé par cette expérience est la fusion des réponses de l'oracle et du classifieur. Il a été décidé de placer le seuil de décision à 0. Or ceci n'est pas forcément le bon seuil du classifieur online. L'oracle va donc dans un premier temps perturber l'apprentissage en fournissant des exemples inadaptés aux besoins du classifieur online car son but est d'asservir le seuil aux alentours de 0.

La table 6.4 contient les caractéristiques de tous les détecteurs au point de F-Mesure maximale. Au niveau du point de fonctionnement optimal au sens de la F-Mesure, les différences entre les classifieurs se font plus faibles. Le classifieur offline reste néanmoins le meilleur des trois.

TABLE 6.4 – Caractéristiques des détecteurs offline et online sur la séquence ViCoMo 1.

	Rappel, (écart type)	Précision	F-Mesure
offline : 2 000 positifs et 8 000 négatifs	0,76 (0,050)	0,83	0,79
online : 45 000 positifs et 50 000 négatifs	0,74 (0,037)	0,81	0,78
online et mise à jour	0,70 (0,026)	0,79	0,74

Ces expériences tendent à montrer que le boosting online n'améliore pas la contextualisation par rapport au boosting offline. L'adaptabilité d'un classifieur online est un facteur de risque important dans le cadre d'un système devant fonctionner de manière autonome.

Les algorithmes d'apprentissage offline semblent plus intéressants pour construire un système de détection de piétons contextualisé. D'autre part, un système de vidéosurveillance n'est pas forcément opérationnel en permanence. La nuit, sans éclairage, il ne voit rien, une gare ou un aéroport peuvent être fermés. Il est alors possible de mettre à profit ce temps non utilisé pour réaliser des apprentissages offline afin de mettre à jour tout le système, ce qui atténue son principal défaut.

6.4 Réglage automatique du seuil de détection

Une fois le classifieur entraîné, il est important de le régler pour en obtenir les meilleures performances. Jusqu'à présent, l'enjeu était de construire le classifieur le plus performant possible en comparant les courbes précision-rappel obtenues. Or dans le cadre d'une exploitation d'un détecteur il est nécessaire de choisir un point de fonctionnement qui réalise un bon compromis entre le rappel et la précision du classifieur. En vidéosurveillance, la caméra étant fixe, la scène traitée est toujours la même. Une fausse détection a de grandes chances d'apparaître très régulièrement, dégradant significativement les performances du système. Réduire le nombre de faux positifs est donc primordial.

Dans le cas du boosting, le réglage d'un classifieur consiste principalement à ajuster le seuil de détection. En effet, les algorithmes de classification utilisés sont symétriques et supposent donc que les exemples positifs et négatifs ont autant d'importance. En conséquence, la non détection d'un piéton est autant pénalisée que la surdétection d'un élément du fond. Cela impose au classifieur de vouloir détecter beaucoup de piétons et le conduit à produire beaucoup de faux positifs. Pour pallier ce défaut, il est généralement admis qu'il faut modifier les seuils de détection du classifieur. La plupart du temps, le seuil optimal est différent de 0.

Dans la suite de cette partie, les principales méthodes pour régler le classifieur seront détaillées. Dans le cadre des algorithmes d'apprentissage symétriques, deux approches existent. L'une locale détermine un seuil pour chaque point de l'image. L'autre globale estime un seuil valable sur toute l'image. Cette dernière méthode est la plus classique et nous proposons une méthode pour estimer un tel seuil de manière automatique dans le cadre de la contextualisation. Enfin une autre possibilité, qui n'a pas été validée au cours de cette thèse consiste à employer un algorithme de classification asymétrique. En pénalisant différemment la non détection des piétons et les faux positifs, il serait peut-être possible de s'affranchir du réglage du seuil.

6.4.1 Estimation de seuils locaux

Le concept des seuils locaux adaptatifs a déjà été évoqué pour l'oracle dans la partie 4.3.5.2. Cette approche utilise un algorithme similaire à ceux de la soustraction de fond pour estimer un seuil en chaque point de l'image. Un tel algorithme pourrait très bien être intégré en sortie du classifieur contextualisé pour déterminer des seuils localement.

Outre le fait de nécessiter des calculs plus importants, ce module risque également d'apporter les défauts de la soustraction de fond. Premièrement, un piéton immobile aura tendance à être appris comme du fond et tendra à disparaître au cours du temps. Cela est acceptable pour l'oracle qui ne cherche pas à détecter de manière exhaustive mais cela ne l'est plus pour le détecteur contextualisé. Néanmoins ce phénomène peut

être atténué si la constante de temps de l'algorithme de soustraction de fond est très grande. Le deuxième problème posé par cette méthode est le bruit généré. Comme dans le cas de la soustraction de fond, il n'est pas rare de voir apparaître des détections sur quelques images seulement. Par exemple, un changement d'éclairage, une ombre ou un objet mobile peuvent provoquer une petite variation du score en un point et celle-ci peut être suffisante pour déclencher une détection. Pour limiter ce problème, [Stalder *et al.*, 2010] couplent cette approche avec un module de tracking. Ce dernier intègre temporellement les détections obtenues et élimine une partie du bruit engendré. Cette approche n'a pas été retenue pour notre système.

6.4.2 Estimation d'un seuil global

6.4.2.1 Principe

Plutôt que de déterminer un seuil en chaque point, nous préférons utiliser une méthode qui estime un seuil de manière globale sur toute l'image ou au moins sur une région de celle-ci (*cf* paragraphe 5.4). Cette approche a plusieurs avantages :

- elle ne nécessite pas de calculs supplémentaires pendant la détection, tout est fait en amont de la phase d'exploitation puisque le seuil est estimé lors de l'initialisation ;
- le seuil est constant au cours du temps, donc les piétons demeurant trop longtemps immobiles ne disparaissent pas des détections.

Le seuil est considéré comme étant optimal pour un classifieur lorsque la F-Mesure est maximale. La F-Mesure établit un compromis entre le rappel et la précision du classifieur lorsque les deux termes ont autant d'importance. Le principal problème pour estimer le seuil est d'évaluer cette F-Mesure. En effet pour connaître le rappel et la précision d'un classifieur sur une séquence, il est nécessaire de disposer de la vérité terrain.

Dans les cas classiques, où le but est de fabriquer un classifieur générique, il existe généralement deux bases labellisées : la base d'apprentissage et la base de test. Cette dernière sert à étalonner le classifieur. Il est ainsi possible d'estimer les propriétés du classifieur et de déterminer le seuil optimal (*cf* figure 6.9). Or dans le cadre de la contextualisation, il n'existe pas de base de test.

Une possibilité pour estimer le seuil de détection du classifieur contextualisé est de choisir comme base de test, la base générique de l'INRIA. Les scores du classifieur pour les exemples positifs et négatifs sont collectés pour trouver le seuil qui sépare au mieux les deux classes au sens de la F-Mesure. Cette approche ne fonctionne pas comme le prouve la figure 6.10. Celle-ci représente la courbe précision-rappel du classifieur contextualisé sur ViCoMo 1 et les deux seuils estimés avec la vérité terrain ou avec la base INRIA, qui sont deux bases labellisées manuellement. Le seuil estimé à l'aide de la base INRIA 17,3 est très supérieur à celui estimé grâce à la vérité terrain 10,6. Dans ces conditions, le rappel est extrêmement bas et l'ensemble de la procédure de contextualisation devient caduc. Cette mauvaise estimation s'explique par le fait que la base INRIA n'est pas une bonne base de test car non représentative des données sur lesquelles doit fonctionner le classifieur contextualisé.

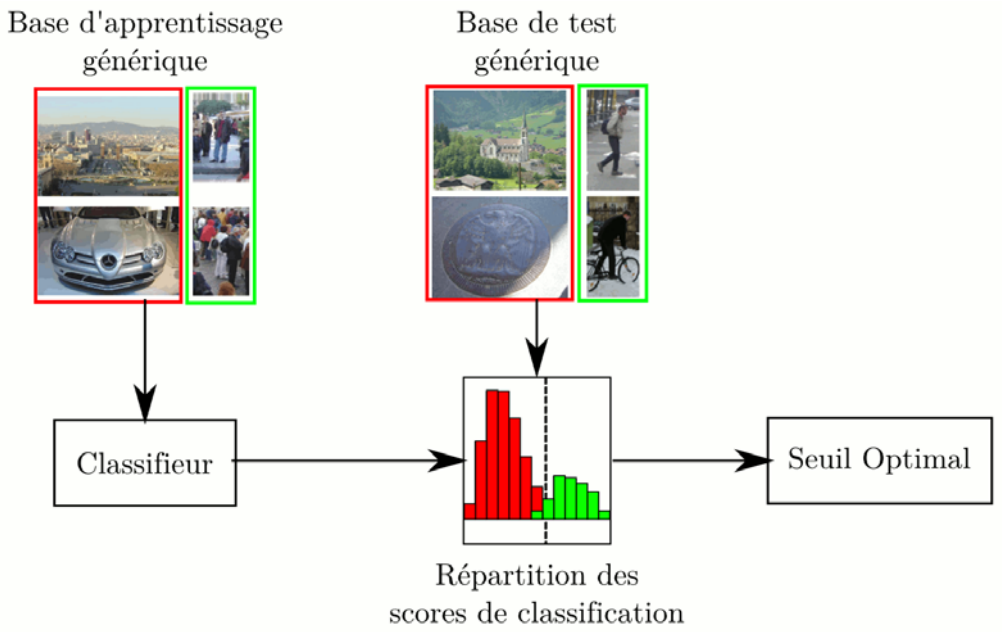


FIGURE 6.9 – Réglage du seuil avec une approche classique. Une base de test sert d'étalonnage pour le classifieur.

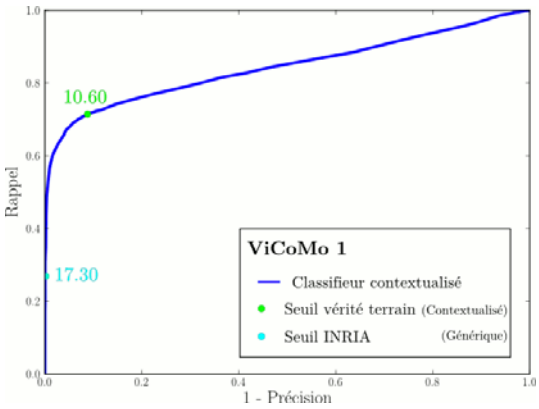


FIGURE 6.10 – Courbe précision-rappel du classifieur contextualisé sur la séquence ViCoMo 1. Les points indiquent les seuils estimés avec la base INRIA (cyan) ou avec la vérité terrain (vert), deux bases construites manuellement.

L'idée est donc de fabriquer une base de test contextualisée en s'aidant de l'oracle (*cf* chapitre 4). Les exemples positifs de cette base de test sont constitués par les exemples fournis par l'oracle. Tous les autres exemples sont étiquetés négativement. En pratique, l'oracle n'est pas parfait et la base est donc bruitée, ce qui peut fortement perturber l'estimation du seuil.

Une fois la base de test constituée, il reste à estimer le meilleur seuil possible. Une première solution serait, à l'instar de [Yao *et al.*, 2012], d'estimer une distribution de probabilité des scores des exemples positifs et négatifs et d'optimiser le seuil pour qu'il minimise l'erreur du classifieur. Cette solution pourrait, peut-être, permettre de s'affran-

chir du bruit présent dans notre base de test, qui est relativement important au niveau du seuil optimal. Ceci est néanmoins difficile en pratique. Les histogrammes 6.12 et 6.13 montrent que la répartition des scores dépend fortement de la séquence traitée. Ces distributions ne peuvent pas non plus être expliquées simplement par des lois de probabilité classiques comme les lois normales, bêta ou gamma et à minima un modèle plus complexe comme un mélange de gaussiennes est requis. La solution retenue ici consiste à traiter directement les scores de classification des exemples de la base de test et à trouver le seuil qui maximise la F-Mesure à l'aide d'une optimisation à une dimension, c'est-à-dire de tester tous les seuils avec un pas de 0,1.

La figure 6.11 présente l'approche retenue pour estimer automatiquement le seuil optimal dans le cas d'un classifieur contextualisé.

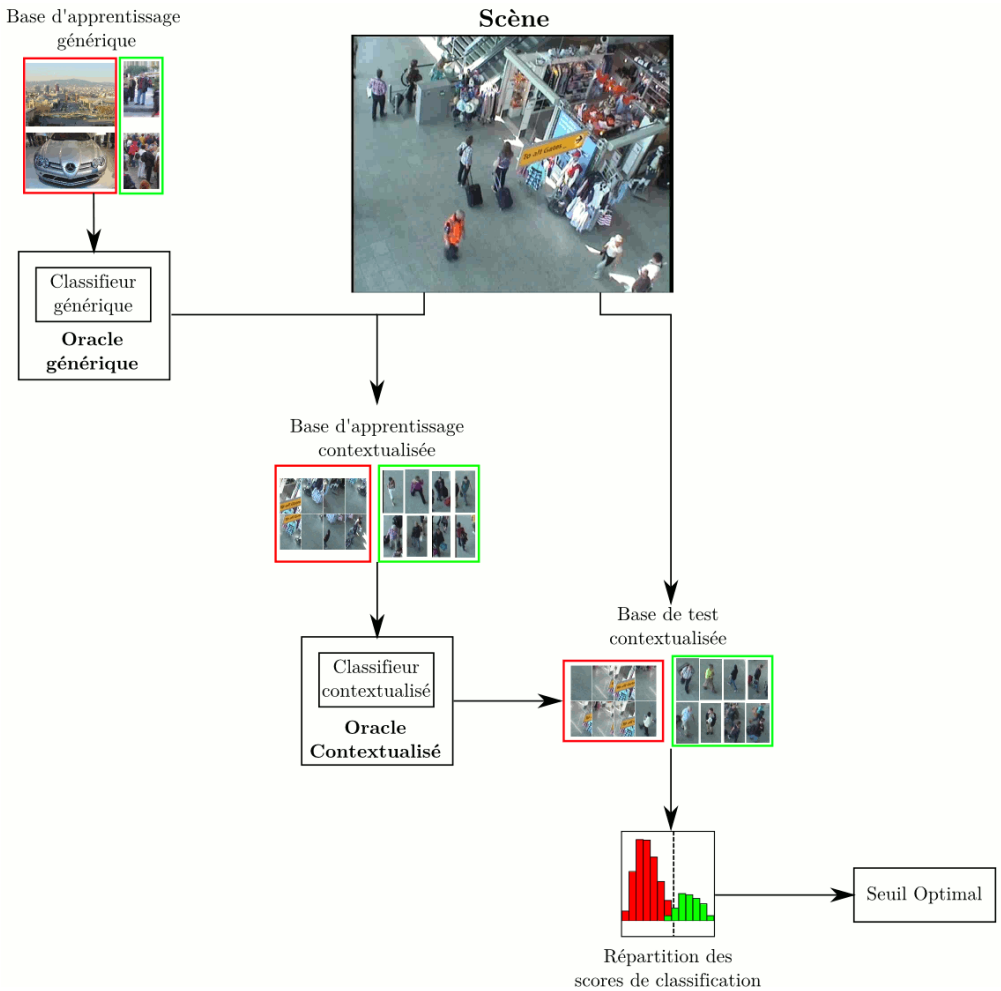


FIGURE 6.11 – Notre méthode de réglage du seuil dans le cas contextualisé. Ne disposant pas d'une base de test, l'oracle contextualisé est utilisé pour labelliser automatiquement une vérité terrain servant d'étalonnage pour le classifieur.

6.4.2.2 Implémentation

Dans cette expérience, pour que la base de test contextualisée soit la plus pertinente possible, il est nécessaire que l'oracle privilégie autant le rappel que la précision pour ses détections. En effet avec un rappel faible comme celui de l'oracle, des exemples positifs vont être placés dans la base des exemples négatifs. Cela conduit à modifier la distribution estimée des scores des exemples positifs et négatifs. Les scores des exemples négatifs auront alors tendance à être surestimés (puisqu'ils incluent des vrais positifs) tandis que ceux des exemples positifs seront en sous effectif (diminuant le poids d'un positif par rapport aux négatifs). Au final, cela tendra à surestimer fortement le seuil de détection.

Pour limiter cet effet au maximum, il a été décidé d'augmenter le rappel de l'oracle. Pour réaliser nos tests, nous avons donc choisi l'oracle 3D (*cf* partie 4.4, page 95) car il possède généralement le meilleur rappel de tous les oracles présentés. En pratique, son rappel est encore un peu trop faible pour cette application. Pour essayer de l'augmenter, deux leviers ont été employés. Le premier consiste à remplacer dans l'oracle le classifieur générique par le classifieur contextualisé. Comme remarqué lors de l'expérience sur le bouclage de la procédure de contextualisation (*cf* partie 6.3.1), cela permet d'augmenter le rappel de l'oracle. Le deuxième changement dans l'oracle concerne le paramètre α de l'équation 4.1 (page 96). Afin de donner plus de poids au score de classification, α est diminué ce qui réduit la part de la soustraction de fond dans la fusion des signaux. En conséquence, un piéton même immobile peut être détecté. Nous avons arbitrairement fixé α à 0,9 au lieu de 1.

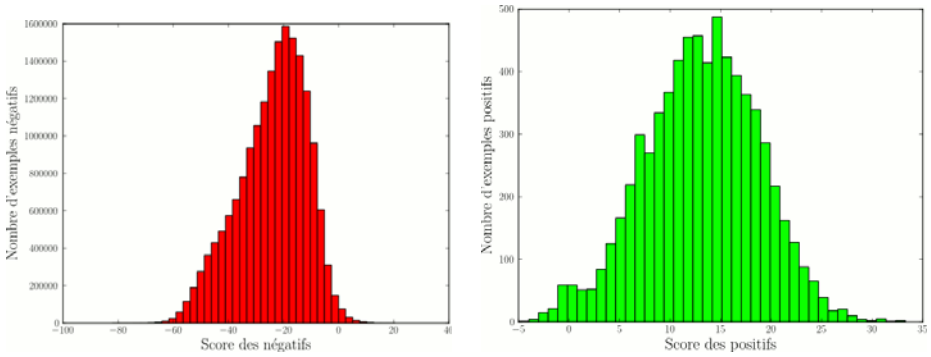
Une fois les positions des exemples fournis par l'oracle connues, il est nécessaire de fabriquer la base de test. Le seuil de détection doit être le plus élevé possible pour limiter au maximum les faux positifs du détecteur. Or la position donnée par l'oracle ne correspond pas forcément à la boîte ayant le score maximal dans la zone correspondant à une détection correcte pour le piéton. Le score retenu pour un exemple positif est donc le score le plus élevé dans la zone admissible autour du piéton. Les boîtes de cette zone correspondent à toutes celles qui sont similaires (*cf* paragraphe 2.3.3.4) à la boîte détectée par l'oracle. Cette stratégie permet aussi d'atténuer légèrement l'ajout d'un élément du fond dans la base des exemples positifs car le score sélectionné sera le plus élevé de la zone ce qui réduit la possibilité d'ajouter des scores très faibles dans cette base.

Enfin, les exemples qui n'ont pas été détectés par l'oracle sont labellisés négativement. Comme expliqué précédemment, il est important de ne pas ajouter de vrais positifs dans la base. Or, lorsque l'oracle trouve un exemple, celui juste à côté peut aussi avoir un score élevé. Pour limiter ce problème nous ne prenons pas en compte les exemples négatifs qui sont proches d'un exemple positif fourni par l'oracle.

6.4.2.3 Évaluations

La méthode d'estimation automatique du seuil a été évaluée sur deux séquences ViCoMo 1 et 2. Elle est comparée avec une procédure d'estimation dite manuelle qui emploie une base construite par un humain, la vérité terrain. L'objectif étant que ces deux approches donnent les résultats les plus proches possible.

Pour la séquence ViCoMo 1, l'oracle générique labellise les 10 000 premières images pour fabriquer le classifieur contextualisé. Ce dernier est ensuite utilisé avec l'oracle contextualisé sur les images 15 000 à 16 000 pour estimer le seuil. La figure 6.12 présente les répartitions des scores négatifs et positifs sur la séquence ViCoMo 1. Il y a évidemment beaucoup plus d'exemples négatifs que d'exemples positifs.



(a) Scores des exemples négatifs (b) Scores des exemples positifs

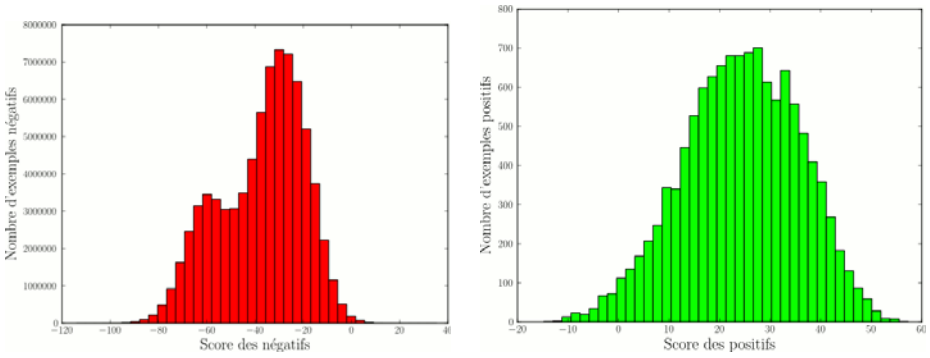
FIGURE 6.12 – Répartition des scores des exemples négatifs et positifs sur la séquence ViCoMo 1.

La table 6.5 montre les seuils estimés automatiquement et manuellement, ainsi que le rappel et la précision en ces points. Les deux dernières lignes du tableau montrent que la méthode d'estimation automatique fonctionne correctement sur cette séquence. Les conclusions sont détaillées à l'issue de la deuxième expérience.

TABLE 6.5 – Seuils optimaux, au sens de la F-Mesure, obtenus avec les deux approches. La première ligne présente les estimations en sortie de la méthode entièrement automatique. La dernière ligne montre celles de la méthode entièrement manuelle. La ligne du milieu indique, pour le seuil estimé automatiquement, les performances calculées d'après la vérité terrain.

ViCoMo 1	Seuil	Rappel	Précision	F-Mesure
Base de l'oracle (automatique)	**11,4**	0,63	0,69	0,66
Base vérité terrain (manuelle)	11,4	0,68	0,95	**0,79**
Base vérité terrain (manuelle)	10,6	0,71	0,91	**0,80**

La même procédure est répétée sur la séquence ViCoMo 2. L'oracle générique labellise les images de 10 000 à 45 000 pour fabriquer le classifieur contextualisé. L'estimation du seuil quant à elle s'effectue sur les 5 000 premières images de la vidéo. La figure 6.13 présente les répartitions des scores négatifs et positifs sur la séquence ViCoMo 2. La table 6.6 montre les seuils estimés à l'aide des bases créées automatiquement ou manuellement, ainsi que le rappel et la précision en ces points. Sur les deux séquences, le seuil estimé est légèrement supérieur à celui estimé avec la vérité terrain. Les courbes précision-rappel des classifieurs contextualisés, ainsi que les points correspondants aux seuils estimés, sont présentés sur la figure 6.14.

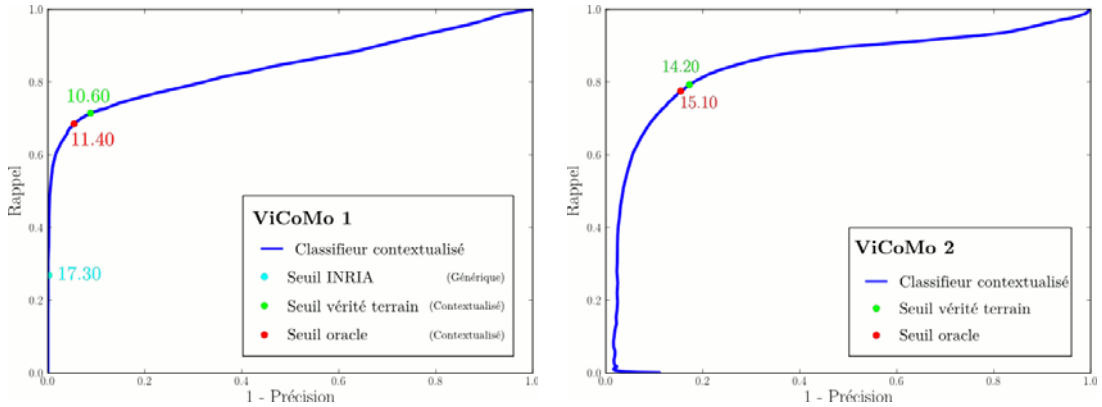


(a) Scores des exemples négatifs (b) Scores des exemples positifs

FIGURE 6.13 – Répartition des scores des exemples négatifs et positifs sur la séquence ViCoMo 2.

TABLE 6.6 – Seuils optimaux, au sens de la F-Mesure, obtenus avec les deux approches. La première ligne présente les estimations en sortie de la méthode entièrement automatique. La dernière ligne montre celles de la méthode entièrement manuelle. La ligne du milieu indique, pour le seuil estimé automatiquement, les performances calculées d'après la vérité terrain.

ViCoMo 2	Seuil	Rappel	Précision	F-Mesure
Base de l'oracle (automatique)	**15,1**	0,79	0,75	0,77
Base vérité terrain (manuelle)	15,1	0.78	0,85	**0,81**
Base vérité terrain (manuelle)	14,2	0,79	0,83	**0,81**



(a) ViCoMo 1 (b) ViCoMo 2

FIGURE 6.14 – Courbes précision-rappel des classifieurs contextualisés sur les séquences ViCoMo 1 et 2. Les points indiquent les seuils estimés avec la vérité terrain (vert, manuelle) ou avec la base de l'oracle (rouge, automatique).

Les lignes 2 et 3 des tables 6.5 et 6.6 indiquent les performances, calculées grâce à la vérité terrain, des classifieurs contextualisés réglés avec un seuil estimé soit automatiquement ou soit à l'aide de la vérité terrain. La comparaison de ces résultats prouve que le réglage du seuil fonctionne correctement sur ces deux séquences. La différence

entre la F-Mesure atteinte par le seuil automatique et celle obtenue par le seuil de la vérité terrain est minime. Néanmoins, comme évoqué dans le paragraphe précédent la méthode d'ajustement du seuil estime généralement un seuil légèrement trop haut. Ce phénomène peut s'expliquer par le fait que l'oracle est imparfait avec un rappel faible, des exemples positifs sont donc placés dans la base des exemples négatifs. Notre méthode a alors tendance à déterminer un point de fonctionnement pour le classifieur contextualisé ayant un rappel un peu plus bas et une précision un peu plus élevée que le point considéré comme optimal.

Pour le moment, la méthode a été utilisée pour estimer un seuil global par scène. Or il est possible de la combiner avec un classifieur par région et d'estimer un seuil différent pour chaque région. La procédure est similaire avec celle employée jusqu'à présent. La seule différence consiste à séparer les scores des exemples positifs et négatifs pour chaque région et à réaliser une optimisation par région.

Nous reprenons les régions définies dans le chapitre 5.4 et réalisons l'expérience sur la séquence ViCoMo 2. La table 6.7 montre les seuils estimés à l'aide des bases créées automatiquement ou manuellement, ainsi que le rappel et la précision en ces points. Les courbes précision-rappel des classifieurs contextualisés de chaque région, ainsi que les points correspondants aux seuils estimés, sont présentés sur la figure 6.15.

Table 6.7 – Seuils optimaux, au sens de la F-Mesure, estimés automatiquement (A) ou manuellement (M) sur la séquence ViCoMo 2.

Région 0

	Seuil	Rappel	Précision	F-Mesure
A	**16,2**	0,73	0,73	0,73
M	16,2	0,68	0,97	**0,80**
M	11,3	0,85	0,91	**0,88**

Région 1

	Seuil	Rappel	Précision	F-Mesure
A	**16,6**	0,93	0,93	0,93
M	16,6	0,90	1.0	**0,95**
M	-3,2	0,98	0,98	**0,98**

Région 2

	Seuil	Rappel	Précision	F-Mesure
A	**33,3**	0,77	0,91	0,83
M	33,3	0,81	1,0	**0,89**
M	-0,30	0,97	0,98	**0,98**

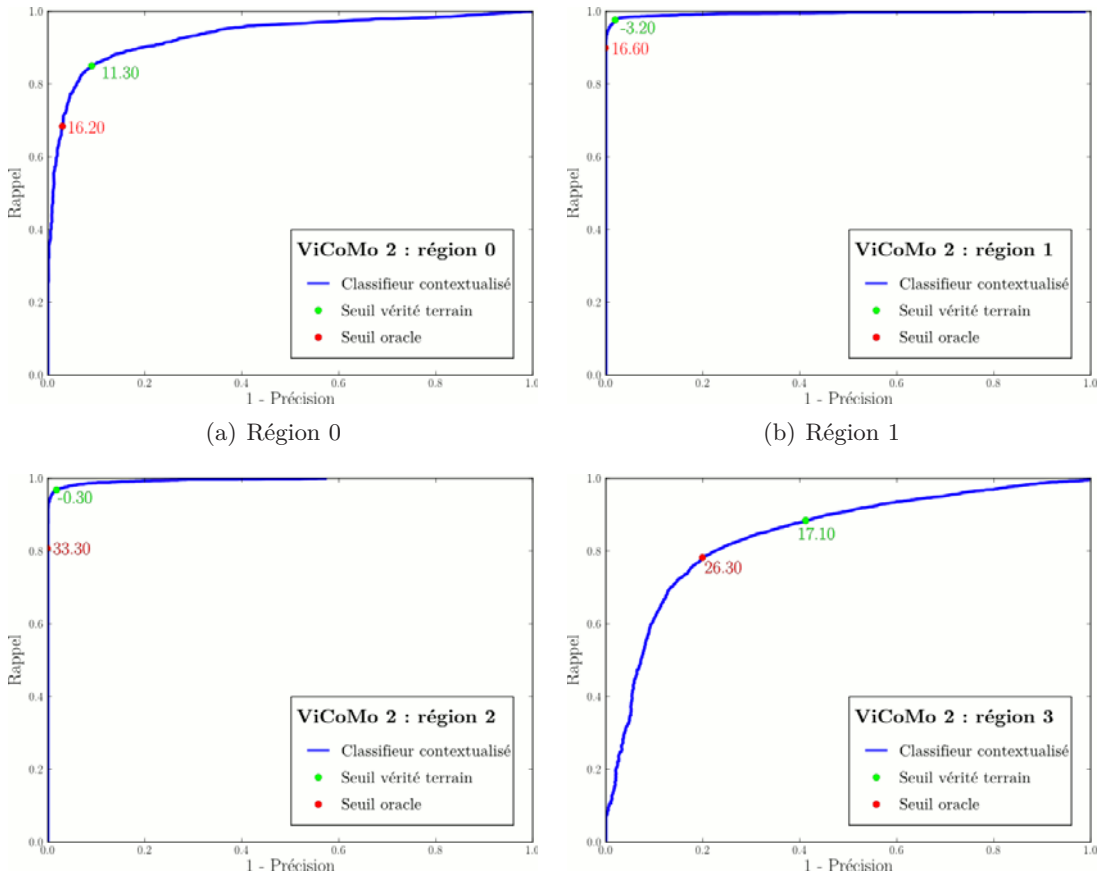
Région 3

	Seuil	Rappel	Précision	F-Mesure
A	**17,1**	0,80	0,92	0,85
M	17,1	0,88	0,58	**0,70**
M	26,3	0,78	0,80	**0,79**

Le réglage d'un seuil par région semble fournir des résultats moins pertinents que pour un seuil global. Là encore des investigations supplémentaires sont nécessaires pour conclure sur les causes de ce phénomène qui peut, peut-être, provenir d'un manque de données pour chaque région.

À noter que comme mentionné lors de la création des zones, les scores fournis par les classifieurs varient indépendamment d'une région à l'autre. Deux seuils parfois très différents peuvent correspondre à des points relativement proches sur la courbe comme dans la région 1 et dans une moindre mesure la région 2.

La région présentant le plus grand écart de performances entre les deux seuils est la région 3. Il s'agit probablement, là encore, d'un problème lié à l'escalier mécanique et à la vérité terrain. En effet cette dernière ne contient pas de piétons à moitié occultés. Or l'oracle les détecte souvent. Il est possible que l'écart entre les deux estimations puisse provenir en partie d'une différence d'appréciation entre l'oracle et la vérité terrain.



(a) Région 0 (b) Région 1

(c) Région 2 (d) Région 3

FIGURE 6.15 – Seuils estimés manuellement (vert) ou automatiquement (rouge) pour des classifieurs contextualisés par régions sur la séquence ViCoMo 2.

La méthode d'estimation automatique du seuil a été validée au travers de ces expérimentations. Les performances du classifieur pour un seuil global obtenu grâce à cette méthode sont souvent très proches de celles calculées avec la vérité terrain. Dans le cas des régions, les écarts sont un peu plus importants. Des investigations supplémentaires sont nécessaires pour mieux comprendre leur origine.

6.4.3 Perspectives : algorithmes d'apprentissage asymétriques

Tous les algorithmes d'apprentissage mentionnés précédemment sont symétriques, c'est-à-dire qu'ils considèrent que l'ensemble des exemples positifs et l'ensemble des exemples négatifs ont la même importance. Or en détection de piétons, seule une faible portion de l'image contient des piétons et il y a donc beaucoup plus d'exemples négatifs que d'exemples positifs. L'hypothèse n'est donc pas valable. Supposer que ces deux groupes ont un risque égal lors de l'entraînement du classifieur revient à dire qu'un faux positif est moins important qu'un faux négatif. En conséquence le détecteur ainsi construit aura tendance à surdétecter.

Historiquement, le poids des exemples a été modifié et le seuil de détection a été

remonté [Viola et Jones, 2001a; Pham et Cham, 2007; Fan et Stolfo, 1999] pour pallier ce problème de manière empirique lors de l'apprentissage. Ainsi il est plus difficile pour le classifieur de détecter des piétons ce qui réduit le nombre de faux positifs. Cette stratégie se retrouve aussi dans l'apprentissage d'une cascade (*cf* paragraphe 2.3.3.2, page 33) pour laquelle le seuil de sortie à chaque étage est ajusté. Récemment [Masnadi-Shirazi et Vasconcelos, 2011] ont proposé des extensions aux principaux algorithmes de boosting. L'idée ici n'est plus de modifier les poids des exemples mais de travailler directement sur la fonction de coût du boosting pour la rendre asymétrique. Les mêmes auteurs ont aussi travaillé sur un algorithme asymétrique pour les SVM [Masnadi-shirazi et Vasconcelos, 2010].

Les algorithmes asymétriques et leurs apports pour la contextualisation n'ont pas été étudiés pendant cette thèse.

6.5 Conclusion

Les chapitres 4 et 5 avaient uniquement pour but de contextualiser le classifieur. L'objectif ici était d'aller plus loin en contextualisant d'autres éléments du détecteur pour construire un système complet de détection de piétons.

Trois aspects importants du détecteur ont été étudiés. Le premier est la stratégie de recherche des hypothèses dans l'image qui est une brique essentielle car elle détermine les boîtes testées par le classifieur. Elle a donc une influence directe sur la rapidité et sur les performances du détecteur. Notre approche repose sur une segmentation de l'image en superpixels qui est ensuite fusionnée avec les exemples issus de l'oracle. Pendant la phase de détection, seules les boîtes, dont le milieu du bord inférieur est dans un superpixel validé, sont examinées.

Le deuxième point abordé : l'intégration de l'oracle et du classifieur contextualisé dans le détecteur et en particulier la possibilité de mise à jour au cours du temps du modèle des piétons. Les expériences, effectuées dans les chapitres précédents, ont toutes montré que la contextualisation améliore les performances du classifieur. Se pose alors la question de savoir, si répéter cette procédure peut être bénéfique à un classifieur déjà contextualisé. Il a donc été décidé de procéder à plusieurs contextualisations successives, en remplaçant à chaque itération le classifieur de l'oracle par celui issu de l'étape précédente. Les résultats montrent que cela n'est pas souhaitable et que le système dérive rapidement.

La mise à jour du classifieur contextualisé a aussi été évoquée. Un algorithme de boosting online, capable d'apprendre de nouveaux exemples en cours de fonctionnement, à été utilisé. Néanmoins, les performances de notre implémentation ne rivalisent pas avec celles du boosting offline et l'intégration d'exemples au fur et à mesure de la détection ne semble pas faire progresser la contextualisation.

Troisièmement, une méthode a été proposée pour résoudre le problème du réglage automatique du seuil de détection. En effet, dans le cas des approches génériques, une base de test est utilisée pour étalonner les sorties du classifieur et ainsi trouver le meilleur seuil possible. Dans le cas de la contextualisation, une telle base n'est pas disponible et l'utilisation d'une base générique ne donne pas des résultats exploitables en pratique.

La solution retenue consiste à modifier légèrement l'oracle pour augmenter son rappel et ainsi collecter de nouveaux éléments provenant de la scène. Ces derniers sont ensuite utilisés de manière classique pour étalonner le classifieur. Cette approche a été validée pour des classifieurs entraînés sur l'image complète et par régions.

La figure 6.16 présente l'architecture proposée pour la réalisation d'un système complet de détection de piétons contextualisé.

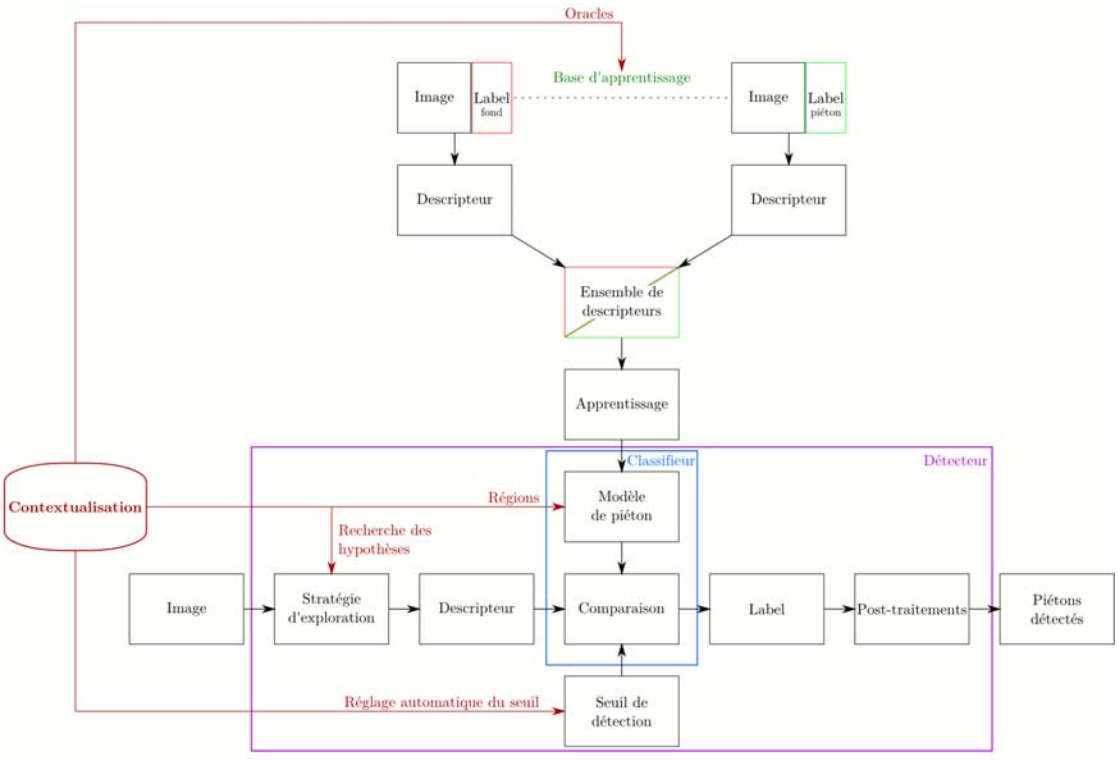


FIGURE 6.16 – Synoptique d'un détecteur de piétons contextualisé.

Enfin, les figures 6.17 et 6.18 montrent de manière visuelle les apports de la contextualisation sur les séquences ViCoMo 1 et 2. La colonne de gauche a été obtenue avec un détecteur générique. Le seuil du classifieur a été calculé grâce à la vérité terrain. Il s'agit donc des meilleures performances possibles, atteignables avec ce détecteur. La colonne de droite utilise un détecteur contextualisé dont le seuil a été estimé avec la méthode automatique. Pour la séquence ViCoMo 2, le détecteur est contextualisé par région, ce qui explique les boîtes de différentes couleurs (une couleur par région). Tous les détecteurs utilisent les zones vides fabriquées manuellement pour ViCoMo 1 et automatiquement pour ViCoMo 2.

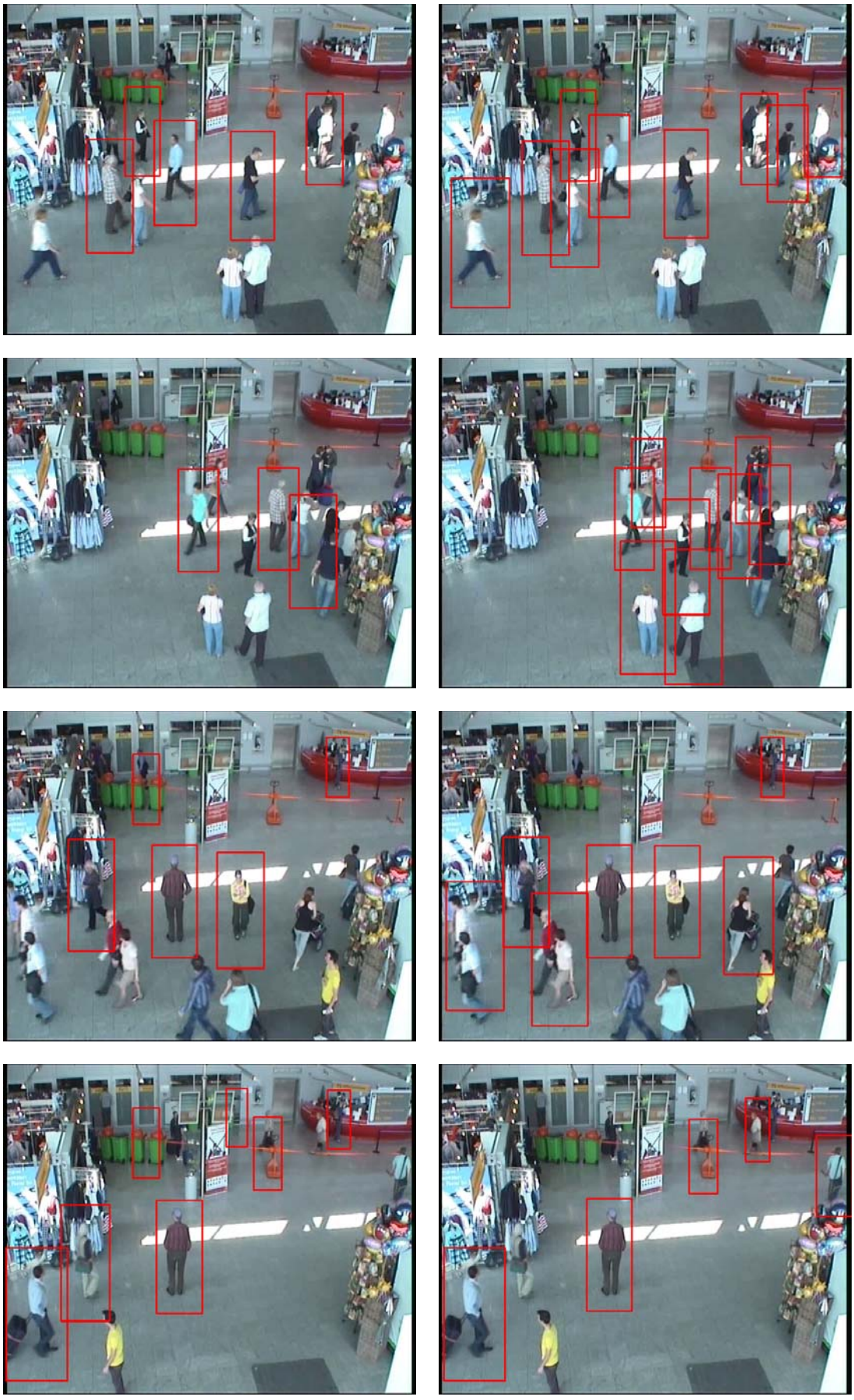


FIGURE 6.17 – Résultats de détection sur la séquence ViCoMo 1. La colonne de gauche correspond à un détecteur générique et celle de droite à notre détecteur contextualisé.

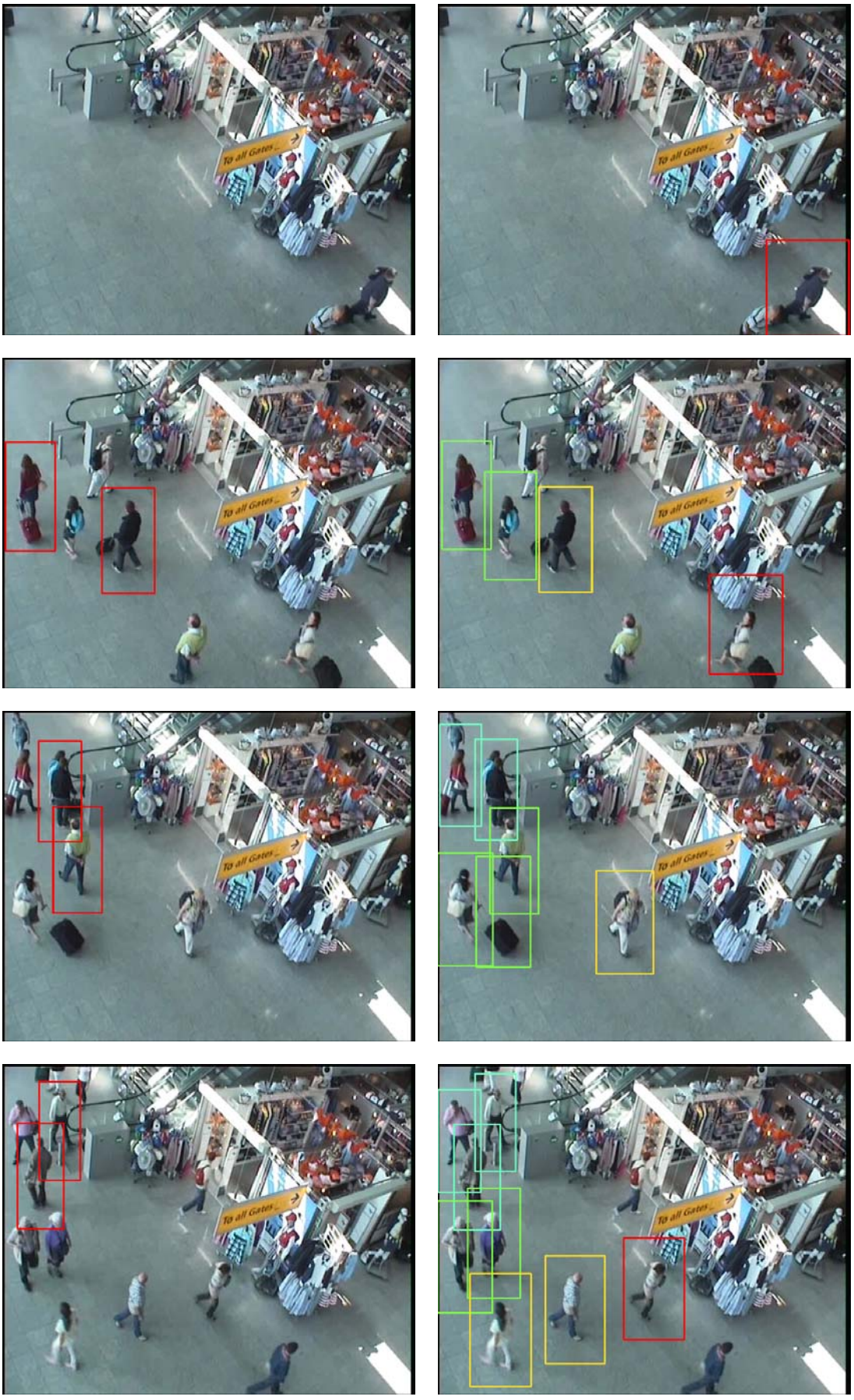


FIGURE 6.18 – Résultats de détection sur la séquence ViCoMo 2. La colonne de gauche correspond à un détecteur générique et celle de droite à notre détecteur contextualisé.

7

Conclusion et Perspectives

Les travaux réalisés au cours de cette thèse ont permis d'améliorer les techniques de « vidéosurveillance intelligente » et plus particulièrement de détection de piétons. Les approches de détection de piétons reposent en majorité sur des outils de reconnaissance de formes basés sur des algorithmes d'apprentissage statistique. Elles sont ainsi capables de reconnaître l'allure générale d'une personne à la condition d'avoir préalablement observé un événement similaire. Cette restriction est fortement contraignante du fait de la grande diversité des situations rencontrées en vidéosurveillance. Il est en effet impossible de collecter puis d'exploiter une base d'apprentissage contenant l'ensemble des situations pouvant être rencontrées par un détecteur.

La contribution majeure de cette thèse a été de proposer un système complet de détection de piétons contextualisé. La contextualisation consiste à enrichir un détecteur en y intégrant directement une connaissance de la scène. Cette approche a l'avantage de dépasser les limites des algorithmes de reconnaissance de formes. Les données d'apprentissage étant spécifiques à la scène, le classifieur est plus simple à construire et offre une réponse mieux adaptée au problème traité. Nos travaux étudient différents aspects de la contextualisation et évaluent leurs apports. Au final, l'ensemble des composants du système sont adaptés à la scène.

De plus, afin de mettre au point un système utilisable à grande échelle, il est important qu'il réponde à certaines exigences fortes. D'une part le processus de contextualisation ne doit pas augmenter les temps de calcul lors de la détection, et d'autre part il doit être automatique afin de pouvoir rapidement déployer le système sur un réseau de caméras.

La première partie du mémoire est consacrée à l'étude de l'état de l'art et aux travaux préliminaires qui ont permis de vérifier l'importance de la contextualisation. La seconde

partie expose les travaux réalisés. Contrairement à un détecteur générique, notre approche utilise une méthode semi-supervisée qui repose sur un oracle. Ce dernier est chargé de labelliser, de manière très précise, un certain nombre d'images de piétons et de fond en provenance de la scène pour construire une base d'apprentissage contextualisée. Celles-ci sont les données indispensables à un algorithme de reconnaissance de formes et ont pour rôle de capter précisément l'apparence des personnes que le système sera amené à rencontrer pendant son fonctionnement. Nos travaux ont abouti à la construction de différents oracles. Tous combinent le signal d'apparence avec un ou plusieurs algorithmes temporels comme la soustraction de fond, le flot optique ou le tracking. Les premiers oracles proposés sont 2D et sont capables de s'adapter aux situations les plus fréquemment rencontrées en vidéosurveillance sans aucune intervention humaine. Le dernier est 3D. Il est plus précis et plus rapide mais nécessite de connaître les paramètres de la caméra pour travailler.

Un classifieur adapté à la scène est ensuite calculé grâce à la base contextualisée collectée par un oracle. Nous avons alors cherché à exploiter au mieux les données brutes fournies par l'oracle lors de la fabrication du classifieur. Ce problème est spécifique à la contextualisation puisque les exemples sont relativement similaires car tous issus de la même scène. Il n'est donc pas forcément pertinent d'entraîner un classifieur avec une base trop importante. D'autre part, malgré toutes les précautions prises lors de la labellisation, l'oracle n'est pas parfait et introduit inévitablement des erreurs dans la base. Plusieurs stratégies de sélection des exemples ont donc été expérimentées. Dans un premier temps nous avons, grâce à la sélection aléatoire des exemples, montré qu'environ 10 000 exemples sont suffisants pour atteindre de bonnes performances et que les exemples négatifs semblent jouer un rôle prépondérant lors de l'apprentissage. D'autres stratégies de choix des exemples basées sur le score, l'apparence ou le suivi des exemples ont aussi été explorées. Toutes ces expériences nous ont amenés à supposer que le plus gros frein à l'augmentation des performances du détecteur venait de notre modèle de représentation des piétons, trop faible pour appréhender la richesse des situations rencontrées par le détecteur. Néanmoins, augmenter la complexité du modèle exige bien souvent des calculs plus importants lors de la phase de détection. Pour ne pas accroître les performances au détriment des temps de calcul, nous avons proposé d'effectuer un découpage de la scène en différentes régions. Un modèle simple et spécifique à chaque région est alors construit. Localement aucun calcul supplémentaire n'est requis puisque la structure des classifieurs est identique à celle d'un classifieur qui ne prend pas en compte les régions. Par contre, d'un point de vue macroscopique, le modèle est plus complexe car il prend en compte une plus grande variété de situations et épouse mieux les spécificités de la scène.

Toutes les techniques mentionnées jusqu'à présent ont pour but de réaliser le meilleur classifieur possible. Or un système de détection complet met en jeu de nombreuses briques logicielles complexes qui peuvent elles aussi bénéficier de la contextualisation. Logiquement nos travaux ont porté sur les autres composants présents dans un détecteur.

Le découpage en régions de la scène a conduit à poursuivre les recherches dans cette voie et à imaginer une méthode simple capable de définir au mieux les hypothèses de recherche du classifieur. Ainsi seules les zones les plus pertinentes sont balayées par le détecteur à la recherche de piétons. Ces zones correspondent aux voisinages des observations collectées par l'oracle.

La mise à jour du classifieur pendant le fonctionnement du système est un point supplémentaire à prendre en compte. Dans un premier temps, le bouclage de la procédure de contextualisation a été étudié pour savoir si cela pouvait être bénéfique. Mais, dans ces conditions, les classifieurs contextualisés obtenus dérivent rapidement et les performances se dégradent. Dans un deuxième temps, nous avons comparé les performances obtenues par notre système stable dans le temps avec celles d'un détecteur online. L'apport de nouveaux exemples durant la phase de détection n'est pas déterminant.

Enfin, la dernière partie de cette thèse propose un réglage automatique du seuil du classifieur contextualisé. Il est en effet inutile de construire automatiquement un classifieur si ce dernier doit être étalonné manuellement à l'aide d'exemples provenant de la scène. Notre méthode repose une nouvelle fois sur l'oracle dont le but est ici de construire une base de test. Celle-ci permet l'estimation d'un seuil de classification qui sépare au mieux les exemples positifs des exemples négatifs.

La contextualisation est un sujet riche et de nombreux points restent à approfondir. Nous avons d'ailleurs mentionné, tout au long du mémoire, plusieurs perspectives pour compléter notre travail.

Une réflexion supplémentaire pourrait être lancée sur les algorithmes d'apprentissage employés. Notre choix s'est initialement porté sur du boosting. La méthode proposée étant complètement indépendante de l'algorithme utilisé, les SVM sont une alternative tout aussi valide. D'autre part, il a été décidé, pour mieux mesurer l'apport de l'approche proposée, de toujours utiliser le même algorithme de boosting. Or les diverses situations rencontrées lors de la contextualisation serait probablement mieux traitées avec des algorithmes spécifiques. Par exemple, la base labellisée par l'oracle contient des erreurs mais aucune mesure pour lutter contre le bruit n'est introduite lors de l'apprentissage. Adaboost est particulièrement sensible à ce phénomène et n'est donc pas forcément adapté dans ce cas. Plusieurs approches existent déjà dans la littérature pour répondre à ce problème. Une piste possible est la limitation des poids des exemples lors de l'apprentissage ou de manière plus théorique l'utilisation d'un algorithme comme MILBoost. Enfin, l'étude des algorithmes asymétriques pourrait peut-être déboucher sur une alternative à notre méthode du réglage de seuil.

La deuxième réflexion possible à l'issue de nos travaux concerne la mise à jour du système. L'hypothèse sous-jacente de notre démarche est que la scène est totalement statique. La contextualisation du détecteur est donc réalisée à l'initialisation et le détecteur contextualisé n'est plus jamais mis à jour. Cependant la scène peut changer. L'éclairage est modifié avec les cycles jour/nuit, certaines zones initialement inaccessibles au public peuvent être ouvertes... L'apport de la contextualisation risque donc de s'estomper au fil du temps. Sur une courte période cela n'a pas d'importance puisque les variations de la scène sont probablement négligeables mais dans le cadre d'un système déployé à grande échelle et devant être opérationnel sur de longues périodes, cela semble une limitation importante. La meilleure réponse à apporter à ce problème dépend de l'application. Dans les cas simples, une mise à jour la nuit ou à intervalles réguliers pourrait peut-être être suffisante. Dans les cas plus complexes, des recherches complémentaires sont nécessaires pour intégrer efficacement les nouvelles informations sans dégrader le système.

Enfin, le point qui aurait probablement le plus d'impact sur nos travaux et pour-

rait améliorer significativement la contextualisation concerne les outils d'analyse et de labellisation de scène (*image parsing*). Aller plus loin que la génération des régions, en comprenant précisément la structure du lieu observé et les relations entre ses composants, aurait des répercussions importantes sur l'ensemble de notre système. Être capable de dire que cet élément est caché par celui-là, que les piétons rentrent par ici, traversent l'image comme cela et ressortent par là, permettrait d'enrichir la détection de piétons et tous les outils qui s'en servent comme le tracking. Cette connaissance agirait par exemple sur la localisation des hypothèses de recherche de l'oracle et du détecteur contextualisé assurant une réduction des erreurs de la base d'apprentissage. Il est probable que le modèle de représentation des piétons puisse aussi en bénéficier, en intégrant par exemple des informations sur la possibilité pour un piéton d'être occulté de telle façon à tel endroit.

Bibliographie

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua et S. Süsstrunk : SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *Pattern Analysis and Machine Intelligence*, 2012. (cité p. 126)

S. Agarwal, A. Awan et D. Roth : Learning to Detect Objects in Images via a Sparse, Part-Based Representation. *Pattern Analysis and Machine Intelligence*, 2004. (cité p. 45)

W. I. Ai et P. Langley : Induction of One-Level Decision Trees. *In Proceedings of the International Conference on Machine Learning*, 1992. (cité p. 30)

D. Arthur et S. Vassilvitskii : K-Means++ : The Advantages of Careful Seeding. *In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007. (cité p. 119)

B. Babenko, P. Dollár, Z. Tu et S. Belongie : Simultaneous learning and alignment : Multi-instance and multi-pose learning. *In RealFaces*, Marseille, France, 2008. (cité p. 34)

J. L. Barron, D. J. Fleet et S. S. Beauchemin : Performance of optical flow techniques. *International Journal of Computer Vision*, 1994. (cité p. 17)

R. Benenson, M. Mathias, R. Timofte et L. V. Gool : Pedestrian detection at 100 frames per second. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2012. (cité p. 124)

M. Black : The Robust Estimation of Multiple Motions : Parametric and Piecewise-Smooth Flow Fields. *Computer Vision and Image Understanding*, 1996. (cité p. 18, 77)

A. Blum et T. Mitchell : Combining Labeled and Unlabeled Data with Co-Training. *In Proceedings of the eleventh annual conference on Computational learning theory*, 1998. (cité p. 55)

A. Bondu et V. Lemaire : État de l'art sur les méthodes statistiques d'apprentissage actif. *RNTI, Numéro spécial sur l'apprentissage et la fouille de données*, 2007. (cité p. 60, 61)

B. E. Boser, I. M. Guyon et V. N. Vapnik : A training algorithm for optimal margin classifiers. *In Fifth Annual Workshop on Computational Learning Theory*, 1992. (cité p. 29)

L. Bottou : Online Algorithms and Stochastic Approximations. *In Online Learning and Neural Networks*. 1998. (cité p. 29)

L. Bourdev et J. Brandt : Robust Object Detection via Soft Cascade. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2005. (cité p. 34)

M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier et L. Van Gool : Online Multiperson Tracking-by-Detection from a Single, Uncalibrated Camera. *Pattern Analysis and Machine Intelligence*, 2011. (cité p. 87, 88, 89, 148)

M. D. Breitenstein, E. Sommerlade, B. Leibe, L. van Gool et I. Reid : Probabilistic Parameter Selection for Learning Scene Structure from Video. *In Proceedings of the British Machine Vision Conference.*, 2008. (cité p. 143)

H. Celik, A. Hanjalic, E. Hendriks et S. Boughorbel : Online Training of Object Detectors from Unlabeled Surveillance Video. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition - Workshops*, 2008. (cité p. 57, 65)

H. Celik, A. Hanjalic et E. A. Hendriks : Towards Unsupervised Learning for Automatic Multi-Class Object Detection in Surveillance Videos. *In Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2009. (cité p. 57, 67)

F. Chang, C. Chen et C. Lu : A Linear-Time Component-Labeling Algorithm Using Contour Tracing Technique. *Computer Vision and Image Understanding*, 2004. (cité p. 16)

O. Chapelle, B. Scholkopf et A. Zien : *Semi-Supervised Learning*. The MIT Press, 2006. (cité p. 53)

Y.-T. Chen, C.-S. Chen, C.-R. Huang et Y.-P. Hung : Efficient hierarchical method for background subtraction. *Pattern Recognition*, 2007. (cité p. 15)

Y. Cheng : Mean Shift, Mode Seeking, and Clustering. *Pattern Analysis and Machine Intelligence*, 1995. (cité p. 37)

T. Chesnais, N. Allezard, Y. Dhome et T. Chateau : Automatic Process to Build a Contextualized Detector. *In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, 2012a. (cité p. 9, 106)

T. Chesnais, N. Allezard, Y. Dhome et T. Chateau : Création automatique d'un détecteur adapté à la scène. *In Congrès Francophone sur la Reconnaissance des Formes et l'Intelligence Artificielle (RFIA)*, 2012b. (cité p. 9, 106)

T. Chesnais, T. Chateau, N. Allezard, Y. Dhome, B. Meden, M. Tamaazousti et A. Chan-Hon-Tong : A Region Driven and Contextualized Pedestrian Detector. *In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, 2013. (cité p. 9, 106, 136)

D. Comaniciu, P. Meer et S. Member : Mean shift : A robust Approach Toward Feature Space Analysis. *Pattern Analysis and Machine Intelligence*, 2002. (cité p. 37)

F. C. Crow : Summed-Area Tables for Texture Mapping. *In Proceedings of the Conference on Computer Graphics and Interactive Techniques*, 1984. (cité p. 21)

W. Dai, Q. Yang, G.-R. Xue et Y. Yu : Boosting for Transfer Learning. *In Proceedings of the International Conference on Machine Learning*, 2007. (cité p. 62)

N. Dalal et B. Triggs : Histograms of Oriented Gradients for Human Detection. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2005. (cité p. 20, 23, 24, 29, 31, 33, 35, 74, 77, 86, 117)

N. Dalal, B. Triggs et C. Schmid : Human detection using oriented histograms of flow and appearance. *In Proceedings of the European Conference on Computer Vision*, 2006. (cité p. 24)

N. Dalal : *Finding people in images and videos*. Thèse de doctorat, Institut National Polytechnique de Grenoble, 2006. (cité p. 37)

J. Davis et M. Goadrich : The Relationship Between Precision-Recall and ROC Curves. *In Proceedings of the International Conference on Machine Learning*, 2006. (cité p. 45)

H. Dee et S. Velastin : How close are we to solving the problem of automated visual surveillance ? *Machine Vision and Applications*, 2008. (cité p. 2, 3)

Y. Dhome, B. Luvison, T. Chesnais, R. Belaroussi, L. Lucat, M. Chaouch et P. Sayd : *Outils d'analyse vidéo : Pour une pleine exploitation des données de vidéoprotection*, chap. Détection d'objets d'intérêt. Hermès Science Publications, 2012. (cité p. 9)

P. Dollár, Z. Tu, P. Perona et S. Belongie : Integral Channel Features. *In Proceedings of the British Machine Vision Conference.*, 2009. (cité p. 117)

P. Dollár, C. Wojek, B. Schiele et P. Perona : Pedestrian Detection : An Evaluation of the State of the Art. *Pattern Analysis and Machine Intelligence*, 2011. (cité p. 19)

P. Dollár, S. Belongie et P. Perona : The Fastest Pedestrian Detector in the West. *In Proceedings of the British Machine Vision Conference.*, 2010. (cité p. 19)

D. Eberly : Perspective Projection of an Ellipsoid, 1999. URL http://www.geometrictools.com/Documentation/PerspectiveProjectionEllipsoid.pdf. (cité p. 35)

S. Y. Elhabian, K. M. El-Sayed et S. H. Ahmed : Moving Object Detection in Spatial Domain using Background Removal Techniques - State-of-Art. 2008. (cité p. 15)

M. Enzweiler et D. M. Gavrila : Monocular Pedestrian Detection : Survey and Experiments. *Pattern Analysis and Machine Intelligence*, 2009. (cité p. 19)

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn et A. Zisserman : The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results, 2009. (cité p. 39)

W. Fan et S. J. Stolfo : Adacost : misclassification cost-sensitive boosting. *In Proceedings of the International Conference on Machine Learning*, 1999. (cité p. 163)

P. Felzenszwalb, D. McAllester et D. Ramanan : A Discriminatively Trained, Multiscale, Deformable Part Model. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2008. (cité p. 33, 124, 125)

P. F. Felzenszwalb et D. P. Huttenlocher : Efficient graph-based image segmentation. *International Journal of Computer Vision*, 2004. (cité p. 126)

M. Fischler et R. Bolles : Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. (cité p. 144)

Y. Freund et R. Schapire : A short introduction to boosting. *The Japanese Society for Artificial Intelligence*, 1999. (cité p. 30, 148)

J. Friedman, T. Hastie et R. Tibshirani : Additive Logistic Regression : a Statistical View of Boosting. *Annals of Statistics*, 1998. (cité p. 30)

K. Fukunaga et L. Hostetler : The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory*, 1975. (cité p. 37)

D. Gerónimo, A. M. López, A. D. Sappa et T. Graf : Survey of Pedestrian Detection for Advanced Driver Assistance Systems. *Pattern Analysis and Machine Intelligence*, 2010. (cité p. 19)

H. Grabner et H. Bischof : On-Line Boosting and Vision. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2006. (cité p. 148, 149, 150)

H. Grabner, P. M. Roth et H. Bischof : Is Pedestrian Detection Really a Hard Task ? *In Proceedings of the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2007. (cité p. 36, 125)

G. Gualdi, A. Prati et R. Cucchiara : Multi-Stage Sampling with Boosting Cascades for Pedestrian Detection in Images and Videos. *In Proceedings of the European Conference on Computer Vision*, 2010. (cité p. 36, 142)

R. I. Hartley et A. Zisserman : *Multiple View Geometry in Computer Vision*. Cambridge University Press, second édn, 2004. (cité p. 35)

D. Hoiem, A. A. Efros et M. Hebert : Geometric context from a single image. *In Proceedings of the International Conference on Computer Vision*, 2005. (cité p. 143)

D. Hoiem, A. A. Efros et M. Hebert : Putting Objects in Perspective. *International Journal of Computer Vision*, 2008. (cité p. 143)

B. K. P. Horn et B. G. Schunck : Determining optical flow, 1981. (cité p. 17)

M. Isard et A. Blake : Condensation-conditional density propagation for visual tracking. *International journal of computer vision*, 1998. (cité p. 88)

O. Javed, S. Ali et M. Shah : Online Detection and Classification of Moving Objects Using Progressively Improving Detectors. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2005. (cité p. 55)

R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova et J. Zhang : Framework for performance evaluation of face, text, and vehicle detection and tracking in video : Data, metrics, and protocol. *Pattern Analysis and Machine Intelligence*, 2009. (cité p. 50)

K. Kim, T. H. Chalidabhongse, D. Harwood et L. Davis : Background modeling and subtraction by codebook construction. *In Proceedings of the International Conference on Image Processing*, 2004. (cité p. 15)

A. Levin, P. Viola et Y. Freund : Unsupervised Improvement of Visual Detectors using Co-Training. *Proceedings of the International Conference on Computer Vision*, 2003. (cité p. 55)

D. D. Lewis et W. A. Gale : A sequential algorithm for training text classifiers. *In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994. (cité p. 61)

R. Lienhart et J. Maydt : An Extended Set of Haar-Like Features for Rapid Object Detection. *In Proceedings of the International Conference on Image Processing*, 2002. (cité p. 21)

J. Liu, R. T. Collins et Y. Liu : Surveillance Camera Autocalibration based on Pedestrian Height Distributions. *In Proceedings of the British Machine Vision Conference.*, 2011. (cité p. 144)

D. G. Lowe : Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 2004. (cité p. 23)

B. D. Lucas et T. Kanade : An iterative image registration technique with an application to stereo vision. *In Proceedings of the International Joint Conference on Artificial Intelligence*, 1981. (cité p. 17)

B. Luvison : *Détection non supervisée d'évènements rares dans un flot vidéo : application à la surveillance d'espaces publics.* Thèse de doctorat, École Doctorale Sciences Pour l'Ingénieur de Clermont-Ferrand, décembre 2010. (cité p. 16)

F. Lv, T. Zhao et R. Nevatia : Camera Calibration from Video of a Walking Human. *Pattern Analysis and Machine Intelligence*, 2006. (cité p. 144)

P. K. Mallapragada, R. Jin, A. K. Jain et Y. Liu : SemiBoost : Boosting for Semi-Supervised Learning. *Pattern Analysis and Machine Intelligence*, 2008. (cité p. 56)

H. Masnadi-shirazi et N. Vasconcelos : Risk minimization, probability elicitation, and cost-sensitive SVMs. 2010. (cité p. 163)

H. Masnadi-Shirazi et N. Vasconcelos : Cost-Sensitive Boosting. *Pattern Analysis and Machine Intelligence*, 2011. (cité p. 163)

L. Mason, J. Baxter, P. Bartlett et M. Frean : Boosting algorithms as gradient descent in function space, 1999. (cité p. 32)

D. Migaud, J.-P. Bayle et J.-M. Bertrand : *Organisation et gestion des forces de sécurité publique*. Cour des comptes, juillet 2011. (cité p. 2, 3)

I. A. Muslea : *Active Learning with Multiple Views*. Thèse de doctorat, University of southern California, 2002. (cité p. 61)

V. Nair et J. J. Clark : An unsupervised, online learning framework for moving object detection. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2004. (cité p. 57, 65, 67)

T. T. Nguyen, H. Grabner, H. Bischof et B. Gruber : On-line boosting for car detection from aerial images. *In Proceedings of the International Conference on Research, Innovation and Vision for the Future*, 2007. (cité p. 56)

T. Ojala, M. Pietikäinen et T. Mäenpää : Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *Pattern Analysis and Machine Intelligence*, 2002. (cité p. 22)

T. Ojala, M. Pietikäinen et D. Harwood : A Comparative Study of Texture Measures with Classification based on Featured Distributions. *Pattern Recognition*, 1996. (cité p. 22)

M. Oren, C. Papageorgiou, P. Sinha, E. Osuna et T. Poggio : Pedestrian Detection Using Wavelet Templates. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 1997. (cité p. 20)

P. Ott et M. Everingham : Implicit Color Segmentation Features for Pedestrian and Object Detection. *In Proceedings of the International Conference on Computer Vision*, 2009. (cité p. 24)

N. C. Oza et S. Russell : Online Bagging and Boosting. *In Artificial Intelligence and Statistics*, 2001. (cité p. 148)

S. J. Pan et Q. Yang : A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010. (cité p. 62)

C. Papageorgiou et T. Poggio : A Trainable System for Object Detection. *International Journal of Computer Vision*, 2000. (cité p. 20)

C. P. Papageorgiou, M. Oren et T. Poggio : A General Framework for Object Detection. *In Proceedings of the International Conference on Computer Vision*, 1998. (cité p. 19, 20)

D. Park, D. Ramanan et C. Fowlkes : Multiresolution models for object detection. *In Proceedings of the European Conference on Computer Vision*, 2010. (cité p. 125)

E. Parzen : On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 1962. (cité p. 37)

M.-T. Pham et T.-J. Cham : Online Learning Asymmetric Boosted Classifiers for Object Detection. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2007. (cité p. 163)

M.-T. Pham, Y. Gao, V.-D. D. Hoang et T.-J. Cham : Fast Polygonal Integration and its Application in Extending Haar-Like Features to Improve Object Detection. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2010. (cité p. 21)

M. Piccardi : Background subtraction techniques : a review. *In Proceedings of the International Conference on Systems, Man and Cybernetics*, 2004. (cité p. 15)

J. C. Platt : Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *In Advances in Large Margin Classifiers*. MIT Press, 1999. (cité p. 74)

X. Ren et J. Malik : Learning a classification model for segmentation. *In Proceedings of the International Conference on Computer Vision*, 2003. (cité p. 126)

M. Rodriguez, J. Sivic, I. Laptev et J.-Y. Audibert : Density-Aware Person Detection and Tracking in Crowds. *In Proceedings of the International Conference on Computer Vision*, 2011. (cité p. 96)

S. Romdhani, P. Torr, B. Schölkopf et A. Blake : Efficient Face Detection by a Cascaded Support-Vector Machine Expansion, 2004. (cité p. 34)

C. Rosenberg, M. Hebert et H. Schneiderman : Semi-Supervised Self-Training of Object Detection Models. *IEEE Workshop on Applications of Computer Vision*, 2005. (cité p. 55)

P. M. Roth, S. Sternig, H. Grabner et H. Bischof : Classifier Grids for Robust Adaptive Object Detection, 2009. (cité p. 36, 52)

P. Roth, H. Grabner, D. Skocaj, H. Bischol et A. Leonardis : Online conservative learning for person detection. *In Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, 2005. (cité p. 57)

C. Rother : A new approach for vanishing point detection in architectural environments. *In Proceedings of the British Machine Vision Conference.*, 2000. (cité p. 144)

D. Rubin *et al.* : Using the sir algorithm to simulate posterior distributions. *Bayesian statistics*, 1988. (cité p. 88)

R. E. Schapire et Y. Singer : Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 1999. (cité p. 30)

B. Settles : Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. (cité p. 60)

H. S. Seung, M. Opper et H. Sompolinsky : Query by committee. *In Proceedings of the ACM Workshop on Computational Learning Theory*, 1992. (cité p. 61)

S. Stalder, H. Grabner, et L. V. Gool : Cascaded Confidence Filtering for Improved Tracking-by-Detection. *In Proceedings of the European Conference on Computer Vision*, 2010. (cité p. 92, 93, 155)

S. Stalder, H. Grabner et L. V. Gool : Beyond Semi-Supervised Tracking : Tracking Should Be as Simple as Detection, but not Simpler than Recognition. *In OLCV 09 : 3rd On-line learning for Computer Vision Workshop, Kyoto, Japan*, 2009a. (cité p. 148)

S. Stalder, H. Grabner et L. V. Gool : Exploring Context to Learn Scene Specific Object Detectors. *In Proceedings of the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2009b. (cité p. 36, 58, 59, 65, 66, 67, 139)

C. Stauffer et W. E. L. Grimson : Adaptive Background Mixture Models for Real-Time Tracking. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 1999. (cité p. 15, 93)

C. Stauffer et W. E. L. Grimson : Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence*, 2000. (cité p. 14, 16)

S. Sternig, P. M. Roth et H. Bischof : Learning of Scene-Specific Object Detectors by Classifier Co-Grids. *In Proceedings of the International Conference on Advanced Video and Signal Based Surveillance*, 2010. (cité p. 36)

M. Tan, L. Wang et I. W. Tsang : Learning Sparse SVM for Feature Selection on Very High Dimensional Datasets. *In Proceedings of the International Conference on Machine Learning*, 2010. (cité p. 29)

K. Toyama, J. Krumm, B. Brumitt et B. Meyers : Wallflower : Principles and practice of background maintenance. *In Proceedings of the International Conference on Computer Vision*, 1999. (cité p. 14)

Z. Tu : Probabilistic Boosting-Tree : Learning Discriminative Models for Classification, Recognition, and Clustering. *In Proceedings of the International Conference on Computer Vision*, 2005. (cité p. 34)

O. Tuzel, F. Porikli et P. Meer : Region Covariance : A Fast Descriptor for Detection and Classification. *In Proceedings of the European Conference on Computer Vision*, 2006. (cité p. 22)

O. Tuzel, F. Porikli et P. Meer : Pedestrian Detection via Classification on Riemannian Manifolds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2008. (cité p. 23)

V. N. Vapnik : *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. (cité p. 29)

A. Vedaldi, V. Gulshan, M. Varma et A. Zisserman : Multiple Kernels for Object Detection. *In Proceedings of the International Conference on Computer Vision*, 2009. (cité p. 34)

P. Viola et M. Jones : Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade. *In Advances in Neural Information Processing Systems*, 2001a. (cité p. 163)

P. Viola et M. Jones : Rapid object detection using a boosted cascade of simple features. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2001b. (cité p. 19, 21, 30, 33, 35, 148)

P. Viola, M. J. Jones et D. Snow : Detecting Pedestrians Using Patterns of Motion and Appearance. *International Journal of Computer Vision*, 2005. (cité p. 19, 20, 30)

P. Viola, J. C. Platt et C. Zhang : Multiple instance boosting for object detection. *In Conference on Neural Information Processing Systems*, 2006. (cité p. 80)

I. Visentini, L. Snidaro et G. L. Foresti : Cascaded Online Boosting. *Journal of Real-Time Image Processing*, 2010. (cité p. 34)

S. Walk, N. Majer, K. Schindler et B. Schiele : New features and insights for pedestrian detection. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2010. (cité p. 117)

M. Wang, W. Li et X. Wang : Transferring a Generic Pedestrian Detector Towards Specific Scenes. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2012. (cité p. 60, 62)

M. Wang et X. Wang : Automatic Adaptation of a Generic Pedestrian Detector to a Specific Traffic Scene. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2011. (cité p. 59, 60, 65, 66, 67, 143)

X. Wang, T. X. Han et S. Yan : An HOG-LBP Human Detector with Partial Occlusion Handling. *In Proceedings of the International Conference on Computer Vision*, 2009. (cité p. 22)

R. Wijnhoven et P. de With : Fast Training of Object Detection using Stochastic Gradient Descent. *In Proceedings of the International Conference on Pattern Recognition*, 2010. (cité p. 29)

B. Wu et R. Nevatia : Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. *In Proceedings of the International Conference on Computer Vision*, 2007a. (cité p. 34, 124)

B. Wu et R. Nevatia : Improving Part based Object Detection by Unsupervised, Online Boosting. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2007b. (cité p. 34, 57, 66, 67, 80)

B. Wu et R. Nevatia : Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors. *In Proceedings of the International Conference on Computer Vision*, 2005. (cité p. 33)

A. Yao, J. Gall, C. Leistner et L. van Gool : Interactive object detection. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2012. (cité p. 156)

J. Yao et J.-M. Odobez : Fast Human Detection from Videos Using Covariance Features. *In Proceedings of the European Conference on Computer Vision - Workshop on Visual Surveillance*, 2008. (cité p. 23)

T. Zhao et R. Nevatia : Bayesian human segmentation in crowded situations. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2003. (cité p. 14)

Q. Zhu, S. Avidan, M. Yeh et K. Cheng : Fast Human Detection using a Cascade of Histograms of Oriented Gradients. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2006. (cité p. 31)

X. Zhu : Semi-supervised learning literature survey. Rapport technique, Computer Sciences, University of Wisconsin-Madison, 2008. (cité p. 53)

Z. Zivkovic : Improved Adaptive Gaussian Mixture Model for Background Subtraction. *In Proceedings of the International Conference on Pattern Recognition*, vol. 2, 2004. (cité p. 77)